A Guide to Integrating with InfoCard v1.0

August, 2005

Authors

Microsoft Corporation
Ping Identity Corporation

Copyright Notice

Abstract

The InfoCard system in the Windows Communications Foundation (WCF) of WinFX allows users to manage their digital identities from various identity providers, and employ them in different contexts where they are accepted to access online services. This Guide describes a model built upon the mechanisms described in [WS-Trust] and [WS-SecurityPolicy] to allow digital identity to be integrated into a user-centric identity framework that promotes interoperability between identity providers and relying parties with the user in control. The mechanisms described in this document provide the framework for an identity metasystem. The interactions between the InfoCard system and a relying party or an identity provider are illustrated to allow others to create identity systems and applications that can use and interoperate with the Windows InfoCard system in WCF. This document is intended to be read alongside the InfoCard Technical Reference [InfoCard-Ref] which provides the normative schema definitions and behaviors referenced by this document.

Status

This draft of the InfoCard Guide reflects what is implemented by the InfoCard system in WCF in the Beta2 release of WinFX. The documented behavior and schema described here are subject to change in the final release of the product.

Table of Contents

1. Introduction

Identity is fundamental to enabling interactions in everyday life. The same is true of the digital world as well where digital identity is fundamental to enabling digital interactions in an interconnected online world. Digital identities are used to authenticate parties to each other in the online world. Knowing, with a high degree of assurance, who one is interacting with is a key element in deciding whether to trust the other party and for what.

Broadly speaking, a ***digital identity*** is a set of *claims* asserted by a *claims authority* about a *principal* that is cryptographically verifiable (see glossary of terms in Appendix B). Claims are typically asserted and communicated in signed security tokens. The claims in security tokens may represent identifying and other personal information about a principal. Users will typically have a portfolio of digital identities analogous to the multiple forms of identities they employ in the physical world – drivers' licenses, other government-issued identity cards, credit cards, company affiliation cards such as frequent flyer cards, etc. The use and acceptance of a digital identity in any given context is usually an intersection of a user's choice to offer an identity based on its appropriateness to the context, and the recipient's choice to accept that identity based on its requirements and willingness to trust the claims authority that is making the claims inherent in the digital identity. Hence it is important to create a system that allows users to employ digital identities issued by different authorities, using different underlying identity technologies, in contexts of their choosing, through a consistent and understandable user interface.

InfoCards presents a model that allows users to manage a portfolio of identities from various authorities, and employ them in different contexts where they are accepted to access online services. It is grounded in the Web services architecture which is based on a suite of specifications that define rich functions that may be composed to meet varied service requirements. A crucial application for these services is to establish a framework in

which consumers of user identities can ask for exactly what they need, and providers of identities can furnish the needed identity with intermediation by the user when appropriate.

In the Web services architecture, digital identities are encoded as *security tokens* containing claims about a user made by an *Identity Provider* (IP) and presented to a *Relying Party* (RP) for user authentication and authorized access to services offered by the relying party. Furthermore, relying parties can express their identity and other security requirements in the form of security policies that can be queried by client applications used by the user to access the services offered by the relying party.

It should be noted that just as claims about users may be asserted by a 3rd party identity providers, some claims could be self-asserted by users acting as their own identity providers as long as the relying party is willing to trust and accept it. It turns out that such self-issued identities are commonly used and find applicability in many everyday online interactions. For example, when users visit an online retailer site and must create new user accounts in order to purchase merchandise from that site, they would normally fill in one or more online forms divulging personal information to the site. In these circumstances, the online retailer site is usually willing to believe and accept the users' *self asserted* personal information to create accounts for them.

To help users organize their various digital identities, the InfoCard model introduces the notion of an *InfoCard* which is an embodiment of a digital identity that the user can visualize, examine and reason about in user interfaces. Each InfoCard corresponds to an identity provider and represents a digital identity of the user issued by that identity provider. Multiple digital identities for the user from the same identity provider would be represented by different InfoCards. Users may have a collection of InfoCards representing the various digital identities they have, some self-issued and others issued by 3rd party identity providers. Note that an InfoCard itself is ***not*** the security token that is used to carry identity claims in Web service protocols, but rather it is an artifact that represents the token issuance relationship of the user with the corresponding identity provider. An actual security token with specific claims can be requested from the identity provider when needed based on the InfoCard. In other words, InfoCards help to provide a concrete visualization of a user's identities on a user interface in digital interactions much like the cards one carries inside one's wallet/purse for everyday physical interactions.

Further, to help users select from their various digital identities in different contexts, the InfoCard model introduces the notion of an *Identity Selector* as an architectural component on the client platform. It provides the processing engine to determine which of a user's InfoCards are applicable in a given interaction as being capable of meeting a relying party's requirements. It also provides a consistent user interface for users to visualize, examine and reason about their digital identities, and select one for use. When an application needs a suitable user identity to satisfy the security requirements of a target service it interacts with, it invokes the identity selector component to obtain the appropriate security token representing the user identity. The identity selector puts users in control of the use of their identities by applications in various contexts.

1.1. InfoCard Model Requirements

The identity selector on a client platform can use InfoCards from any identity provider of the user's choice, and offer those identities to relying parties under user control in contexts where it is appropriate. The identity selector interoperates with the identity provider for an InfoCard using open protocols. The primary goal of this Guide is to document and describe how a relying party expresses its identity requirements to a client such that the identity selector can process them, and how the identity selector interacts with identity providers to obtain security tokens that fulfill those requirements. We hope that this will enable any

identity provider or relying party to interoperate using the InfoCard model for the purpose of identity-based Web service interactions.

The following list identifies the key driving requirements for the InfoCard model:

- Enable use of digital identity in the form of claims in security tokens as authentication and/or authorization data using Web service mechanisms.
- Allow users flexibility in their choice of digital identities they wish to employ, and put users squarely in control of the use of their identities in digital interactions.
- Support cryptographically verifiable but human-friendly identification of the recipients of a user's digital identities.
- Enable interoperability with identity providers and relying parties using open protocols to allow an identity ecosystem to thrive.
- Remain agnostic of specific security token types and identity mechanisms so as to effectively be a conduit for identity flow under user control between identity providers and relying parties.
- Safeguard user privacy by providing privacy-friendly identity mechanisms to help thwart tracking of users' online behavior and unsolicited collusion.
- Provide a simple identity provider to allow users to construct and employ self-issued identities in Web service interactions when acceptable.

1.2. InfoCard Identity Usage Model

This section describes the overall model for using digital identity in the form of security tokens for authentication, authorization and access control, or any other online purpose. The model described here builds on the foundations of WS-Trust, WS-SecurityPolicy and WS-MetadataExchange Web service specifications. The mechanisms described in these specifications provide the basis for expressing identity requirements, and requesting and obtaining security tokens that satisfy those requirements.

As described in [WS-SecurityPolicy], a Web service can express in its policy a requirement for security tokens with a required set of claims in order to process incoming requests. A client system or application acting as the user agent can query and learn the Web service policy using the mechanisms described in [WS-MetadataExchange] prior to requesting service. Upon evaluating the Web service policy, if the client system does not have the necessary security token(s) to satisfy the required claims, it may use the mechanisms described in [WS-Trust] to acquire the necessary security tokens from suitable identity providers and present them to the Web service. The InfoCard model builds on this foundation by describing how these mechanisms are combined to enable rich expression of identity requirements and mechanisms to fulfill them in order to promote interoperability between identity providers and relying parties under user control.

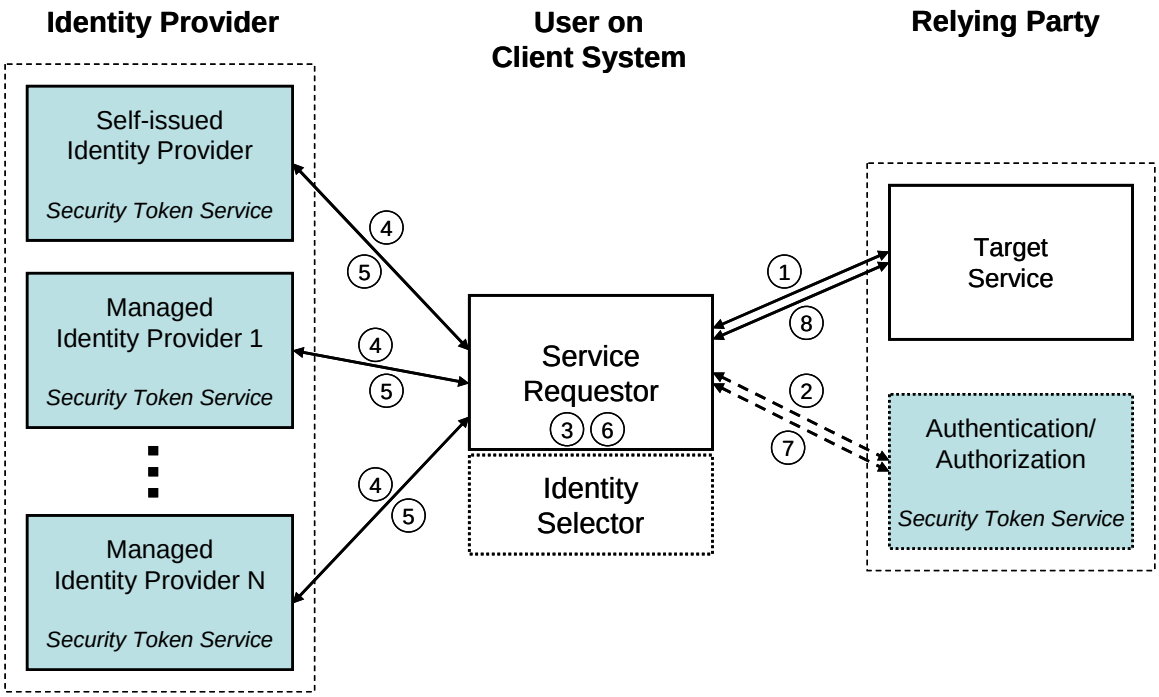

Figure 1 InfoCard Interaction Model

Figure 1 above illustrates the InfoCard interaction model for using digital identity in the form of security tokens in a simple canonical scenario involving a *service requester* in the form of a client application running on a client system, a *relying party* in the form of a target service that the user wishes to access through the service requester application, and one or more *identity providers* that can issue security tokens. The target service may optionally maintain profile information about a user, but that is distinct from the digital identity employed by the user to gain access to the service.

The target service may **optionally** delegate the function of authenticating and validating a user's identity to an associated service shown as the "Authentication/Authorization Security Token Service" in the figure. Note that this is purely a configuration choice for deploying the service and **not** a required configuration. The target service could just as well perform those functions itself instead of delegating. If that were the case, then the extra hop represented by steps 2 and 7 would be absent in Figure 1. Such configuration choices in service deployment can be suitably reflected in the service's security policy causing the appropriate orchestration of security token flows to occur between the service requester and the target service without affecting the basic wire protocol.

The user, interacting through the identity selector on the service requester, may have identities issued by one or more identity providers. Each such digital identity of the user is represented by an *InfoCard* that the identity selector on the service requester system can process. An InfoCard endows the identity selector with the ability to request and obtain security tokens from the corresponding identity provider when the user selects that digital identity for use in a given interaction context.

Each identity provider, shown as "Managed Identity Provider 1 through N" in the figure, runs a Security Token Service (STS) to which a requester can submit security token requests. The security token service can issue security tokens containing the requested claims after the requester has provided suitable proof of authentication as required by the identity provider's security policy. Note that a simple identity provider, shown as the "Self-

issued Identity Provider" in the figure, may be provided by a client platform in the InfoCard model to allow users to issue self-issued security tokens.

Let us assume that the user has previously obtained one or more "InfoCards" from various identity providers and made his InfoCard collection available to the identity selector on the service requester system. A possible sequence of actions that occurs in the model depicted in Figure 1 where the user wants to access a target service through the service requester as the user agent is as follows:

1. The service requester application obtains the security policy of the target service using metadata exchange (the mechanisms described in [WS-MetadataExchange]). The target service policy requires that the requester obtain and present a security token issued by the delegate Authentication/Authorization STS (There is a trust relationship between the target service and its delegate STS).
2. (OPTIONAL) Before the service requester can obtain a security token issued by the Authentication/Authorization STS, it first obtains the security policy of the STS using metadata exchange. The policy requires that the requester obtain and present a security token with specific claims issued by a specified identity provider (which could be the self-issued identity provider).
3. The service requester application requests the identity selector component to produce a security token that can satisfy the target service policy. The identity selector displays the matching InfoCards which can satisfy the target service policy, and the user selects and approves one for use.
4. Before the identity selector on the service requester can request a security token from the identity provider STS for the selected InfoCard, it first obtains the security policy of the identity provider STS using metadata exchange to determine the security binding to use for the message channel.
5. The InfoCard specifies the required authentication mechanism to request security tokens. The identity selector authenticates to the identity provider STS using an acceptable user credential as specified by a credential selector in the InfoCard. It requests a security token of the appropriate type with the required claims as specified by the target service and receives the token using the mechanisms described in [WS-Trust].
6. The identity selector hands the requested security token to the service requester application.
7. (OPTIONAL) The service requester application presents the token and provides proof-of-possession to the Authentication/Authorization STS, and requests a security token for the target service using the mechanisms of [WS-Trust].
8. The service requester application presents the token obtained in step 6 (or optionally step 7) and provides proof-of-possession to the target service to gain access.

Figure 2 below shows the interactions between the participating entities and the sequence of message flows between them. For convenience, the optional Authentication/Authorization STS is omitted from the interactions shown, and the target service is assumed to be the relying party for the security tokens presented by the service requester. The interaction sequence illustrated in this figure provides the framework for the InfoCard model details described in the remainder of this document.

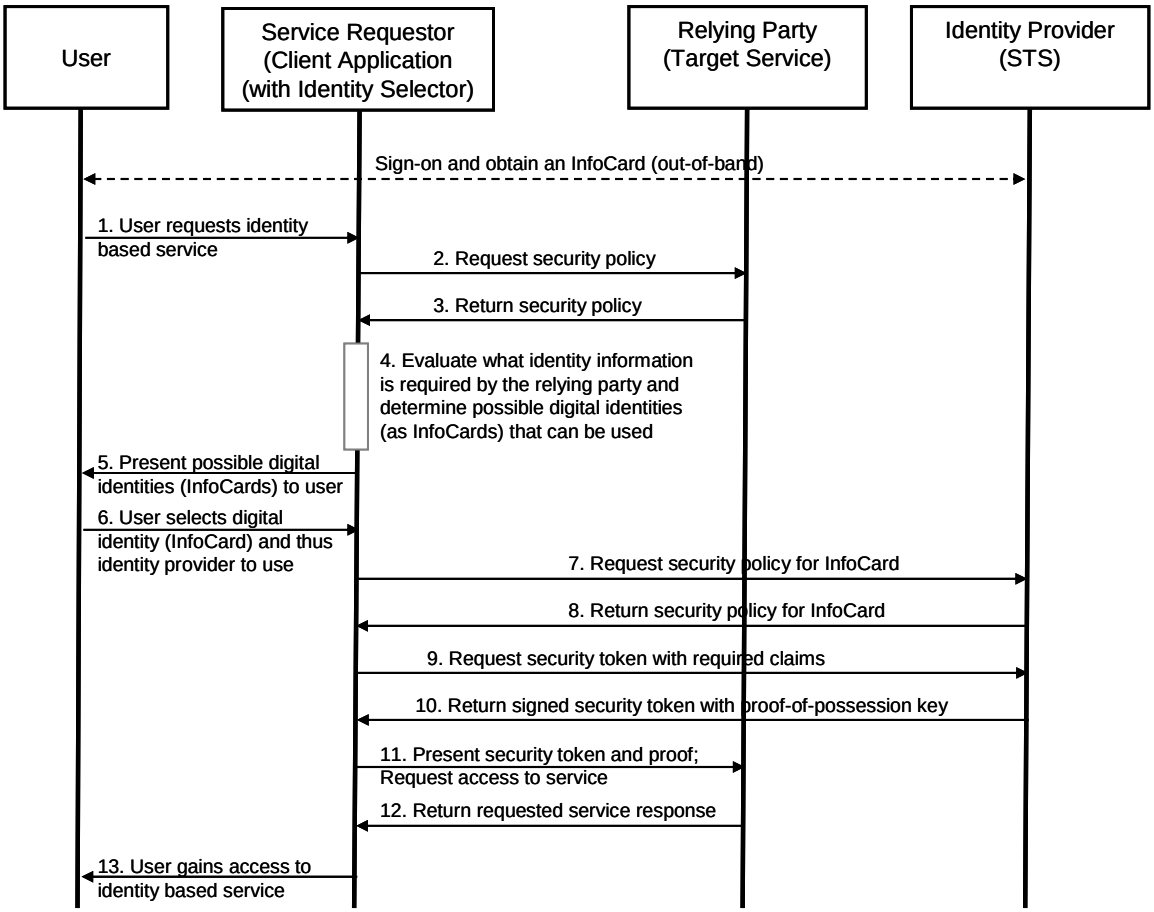

Figure 2 Message flow sequence for InfoCard based interactions

In summary, this Guide builds on the mechanisms described in [WS-Trust], [WS-SecurityPolicy] and [WS-MetadataExchange] to allow digital identity to be integrated into a token issuance and consumption framework that promotes interoperability between identity providers and relying parties with the user in control. In the sections that follow, additional details are provided on the interactions between the service requester and the other two constituents of this model, namely relying parties and identity providers.

1.3. Example of InfoCard Interaction Model

Let us illustrate the InfoCard based interactions described in the previous section with a real world example. John Kane is an employee of Fabrikam, Inc. Fabrikam has a partnership with Blue Yonder Airlines for making travel arrangements for its employees and purchasing tickets at specially discounted prices. Fabrikam has issued all its frequent traveler employees, including John, InfoCards to prove that they are employees of Fabrikam. It also runs a STS at the address *http://fabrikam.com/employee/sts* which issues security tokens for the issued InfoCards. Fabrikam has also given those employees smart cards to use as strong two-factor credentials for authenticating to the employee STS when using their InfoCards.

Employees of Fabrikam use a special travel reservation smart client application for requesting travel arrangements from Blue Yonder Airlines which runs a travel portal and airline reservation service at the address *http://www.blueyonderairlines.com/travel*. When John runs the smart client reservation application on his personal computer to make travel reservations with Blue Yonder Airlines, the following sequence of interactions occur.

- The travel reservation client application obtains the security policy of the airline reservation service at *http://www.blueyonderairlines.com/travel* using metadata exchange. The travel portal service policy requires that the client application submit a security token issued by the user's employer STS, namely the Fabrikam STS at *http://fabrikam.com/employee/sts* (There is a trust relationship between the airline reservation service and each of the partner company STS with which it federates).
- The travel reservation client application requests the identity selector component on John's personal computer to produce a security token that can satisfy the reservation service policy. The identity selector displays the matching InfoCards, namely the InfoCard given to John by his employer, and John selects and approves it for use.
- The identity selector on John's personal computer then obtains the security policy of John's employee STS at *http://fabrikam.com/employee/sts* using metadata exchange to determine the security binding to use for the message channel for requesting the security token.
- The employer issued InfoCard selected by John specifies the required authentication mechanism to be X509 certificate based, and the credential selector in the InfoCard provides a hint for John to insert his smart card given to him by his employer. The identity selector prompts John to insert his corporate smart card into the reader and enter his PIN.
- The identity selector now authenticates to the Fabrikam employee STS at *http://fabrikam.com/employee/sts* using the X509 certificate from John's smart card, and requests a security token with the required claims specified by the airline reservation service. Upon successful authentication, it receives the security token.
- The identity selector then hands the requested security token to the travel reservation application running on John's personal computer.
- The travel reservation application running on John's personal computer then presents the token obtained from the identity selector and presents it to the travel portal service along with proof-of-possession to gain access.
- Now, John can look at possible travel choices and request reservations.

2. Using This Document

In this document we will cover what you need to know and the steps you need to take to support InfoCards either as a relying party or as an identity provider. Following is a brief navigational summary of the parts of this document that are pertinent for each role.

In order to support InfoCards as a *relying party*, you will need to:

- Support identification of your organization using X509v3 certificates with logotypes, whenever possible, to allow end users to clearly identify who they are dealing with. This is described in Section 3.1.
- Support expressing the security requirements of your service, including its security token requirements, using the policy assertions described in [WS-SecurityPolicy]. This is described in Section 3.2.
- Support the retrieval of your service metadata, including its WSDL and policy, using the mechanism described in [WS-MetadataExchange]. This is briefly described in Section 3.3.

- Support the submission of security tokens bound to application mesasges by a service requester using the mechanisms specified in [WS-SecurityPolicy]. This is briefly described in Section 3.4.
- Optionally, when appropriate, support self-issued security tokens and the strong cryptographic keys in such tokens as user credentials instead of passwords. This is described in Appendix A.

In order to support InfoCards as an *identity provider*, you will need to:

- Support issuing InfoCards to your users using the format and mechanism described in Sections 4.1 and 4.2.
- Support the mechansim described in [WS-Trust], using the *RequestSecurityToken* and *RequestSecurityTokenResponse* protocol messages, for issuing security tokens based on an InfoCard. You can, however, issue any type of security token that is acceptable to a relying party since the identity selector on the service requester is token agnostic. This is described in Section 3.4.1, and the message exchanges are detailed in Section 5.
- Support the specific extensions and usage restrictions of WS-Trust protocol elements required by the InfoCard model. This is described in Sections 4.3 and 4.4.
- Support expressing the security requirements of your security token service using the policy assertions described in [WS-SecurityPolicy], and support the retrieval of that policy using the mechanism described in [WS-MetadataExchange]. This is described in Section 5.1.
- Support one or more of the credential mechanisms described in Section 5 to allow users to authenticate to your security token service. When appropriate, instead of passwords, support self-issued security tokens and the strong cryptographic keys in such tokens as user credentials. This is described in Section 5.6.

For brevity of the examples used for illustration in this document, we list here in Table 1 the XML namespaces and corresponding prefixes used throughout the document.

Table 1: Prefixes and XML namespaces used in this document

Prefix	XML Namespace	Reference(s)
S	http://www.w3.org/2003/05/soap-envelope	SOAP 1.2 [SOAP 1.2]
xs	http://www.w3.org/2001/XMLSchema	XML Schema [Part 1, 2]
ds	http://www.w3.org/2000/09/xmldsig#	XML Digital Signatures
ic	http://schemas.microsoft.com/ws/2005/05/identity	InfoCard Technical Reference [InfoCard-Ref]
wsid	http://schemas.microsoft.com/windows/wcf/2005/09/addressingidentityextension	Identity Extension for Web Services Addressing [Addressing-Ext]
wsx	http://schemas.xmlsoap.org/ws/2004/08/mex	WS-MetadataExchange [WS-MetadataExchange]
wsa	http://schemas.xmlsoap.org/ws/2004/08/addressing	WS-Addressing [WS-Addressing]

wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd	WS-SecurityUtility
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.1.xsd	WS-Security Extensions [WS-Security]
wst	http://schemas.xmlsoap.org/ws/2004/04/trust	WS-Trust [WS-Trust]
wsp	http://schemas.xmlsoap.org/ws/2004/09/policy	WS-Policy [WS-Policy]
sp	http://schemas.xmlsoap.org/ws/2005/07/securitypolicy	WS-SecurityPolicy [WS-SecurityPolicy]

3. Relying Party Interactions

This section describes the general framework used by a relying party Web service for specifying and conveying its security token requirements as well as its own identity to a service requester. The relying party Web service may be an application service or a delegate STS supporting the application service as shown in Figure 1. For brevity, the relying party STS shown as “Authentication/Authorization STS” in the figure will be referred to as “RP/STS” in the remainder of this section.

At a high-level, security policy mechanisms described in [WS-SecurityPolicy] is used by a relying party service or a RP/STS to specify its security token requirements and how messages should be secured on a communication path. A service requester can obtain the security policy from the relying party ahead of time or just-in-time using the mechanisms specified in [WS-MetadataExchange] before sending any application request messages. In general, the service requester is required to obtain the required security tokens from the appropriate issuing authorities to satisfy the relying party policy and submit each token along with proof of possession. The service requester submits the required tokens by binding them to application request messages using binding-specific mechanisms described in [WS-SecurityPolicy].

3.1. Identifying the Relying Party

One of the driving requirements of the InfoCard model, and thus the InfoCard system implemented in Windows, is to support cryptographically verifiable but human-friendly identification of the recipients of a user’s digital identities. When a relying party requires that a user’s digital identity be submitted in the form of a security token containing claims, the user needs to have a reliable means to identify the relying party to make a trust decision of whether to release her digital identity or not. This requires that the identity of the relying party be conveyed to the service requester in a form that can be authenticated by the requester yet presented to the user in a human-friendly manner for making trust decisions. Furthermore, it is important that the conveyed identity of the relying party be that of the organization or enterprise represented by the relying party Web service. Users make a conscious choice of whether or not to trust the actual organization or enterprise behind the Web service with their digital identities, ***not*** a specific service end-point.

Given the motivation described above, we recommend using X509v3 certificate for an organization (as opposed to SSL server certificates) with a unique subject identifier identifying the organization. Furthermore, we recommend using certificates with logotypes for the issuer organization and subject organization [RFC 3709] for the purpose of identifying the relying party organization. Much of the information contained in digital certificates is appropriate and effective for machine processing; however, this information is

not suitable for a corresponding human trust and recognition process. The use of logotypes is aimed at simplifying the human interpretation of the certificate content and helping the user's decision to trust the subject organization.

The next question is how is this organizational identity conveyed to the service requester and surfaced to the user? Endpoint references, defined in WS-Addressing, convey the information needed to reference a Web service endpoint. The InfoCard model uses the `Identity` element defined in [Addressing-Ext] to add identity information to an endpoint reference. This identity extension for an endpoint reference should be used to convey the identity of the organization behind that endpoint.

Here is an example of an endpoint reference augmented with identity data in the form of an X509v3 certificate:

```
<wsa:EndpointReference>
  <wsa:Address>http://wh1.fabrikam123.com/Purchasing</wsa:Address>
  <wsid:Identity>
    <ds:KeyInfo>
      <ds:X509Data>
        <ds:X509Certificate>...</ds:X509Certificate>
      </ds:X509Data>
    </ds:KeyInfo>
  </wsid:Identity>
</wsa:EndpointReference>
```

Security tokens returned by the identity selector to the service requester application for submission to the target service will be encrypted by the identity selector (if not already done so by the IP) to the key in the organizational certificate conveyed in the endpoint reference. Since the user evaluates and approves the identity of the organization in the endpoint reference as the recipient of his digital identity, encrypting the tokens as such guarantees that only the entity approved by the user can examine the content of the security tokens.

Other forms of organizational identity and reputations for organizations are possible, and can easily be accomodated in the InfoCard model in the future.

3.2. Expressing Token Requirements of Relying Party

This section describes the mechanisms available to a relying party for specifying its security token (i.e. user identity) requirements as a prerequisite to providing service. The policy assertions and parameters described here are those already present in [WS-SecurityPolicy] or extended by [InfoCard-Ref] where needed.

A relying party specifies its security token requirements as part of its security policy using the primitives and assertions described in [WS-SecurityPolicy]. The primary construct in the security policy of the relying party used to specify the type and claims content of security tokens issued by an identity provider is the `<sp:IssuedToken>` policy assertion. The basic form of the issued token policy assertion as defined in [WS-SecurityPolicy] is as follows.

```
<sp:IssuedToken sp:Usage="xs:anyURI" sp:IncludeToken="xs:anyURI" ...>
  <sp:Issuer>
    <wsa:EndpointReference>...</wsa:EndpointReference>
  </sp:Issuer>
  <sp:RequestSecurityTokenTemplate>
    ...
  </sp:RequestSecurityTokenTemplate>
  <wsp:Policy>
    ...
  </wsp:Policy>
```

```
  ...
</sp:IssuedToken>
```

The following subsections describe the use of special parameters and policy assertion elements added by [InfoCard-Ref] as extensions to the sp:IssuedToken policy assertion that convey additional instructions to the service requester.

3.2.1. Issuer of issued tokens

The optional sp:IssuedToken/sp:Issuer element contains an endpoint reference for the issuer for the required token. More specifically, it should contain the endpoint reference of an identity provider STS that can issue the required token.

A relying party can indicate a specific issuer (*i.e.* identity provider STS) for the required token.

A relying party can also omit the sp:Issuer element or specify it as an empty element to indicate that the service requester should determine the appropriate issuer for the required token with help from the user if necessary. The relying party can evaluate the security token submitted by the requester in response to this policy assertion to make the final determination of whether the token is acceptable. The ability to leave the issuer unspecified in this policy assertion is useful in circumstances where the relying party cannot publicize which identity providers it is willing to accept for confidentiality reasons. For example, an enterprise that federates with other business partners from whom it can accept user identities may have a need to keep confidential who its business partners are.

A relying party can indicate its willingness to accept a self-issued security token from the user by using the special URI below (defined in [InfoCard-Ref]) as the value of the wsa:Address element for the issuer within the sp:IssuedToken policy assertion.

```
http://schemas.microsoft.com/ws/2005/05/identity/issuer/self
```

Following is an example of using this URI within an issued token policy.

Example:

```
<sp:IssuedToken ...>
  <sp:Issuer>
    <wsa:EndpointReference>
      <wsa:Address>
        http://schemas.microsoft.com/ws/2005/05/identity/issuer/self
      </wsa:Address>
    </wsa:EndpointReference>
  </sp:Issuer>
  ...
</sp:IssuedToken>
```

3.2.2. Type of proof key in issued tokens

The default bahavior for an identity provider is to issue a *symmetric key* token if no explicit key type is specified in the token request (as specified in [WS-Trust]). The identity selector, when requesting a security token from an identity provider, communicates the requested key type exactly as specified by the relying party in its issued token policy.

If an explicit key type is not specified by the relying party in its issued token policy, then a symmetric key token will be issued by the identity provider. However, a relying party can explicitly request the use of an *asymmetric* or *symmetric* key in the issued token by using the wst:KeyType element within its issued token policy assertion. The following example illustrates the use of this element in the relying party's security policy to request an asymmetric key in the issued token.

Example:

```
<sp:IssuedToken>
  <sp:RequestSecurityTokenTemplate>
    <wst:KeyType>
      http://schemas.xmlsoap.org/ws/2004/04/trust/PublicKey
    </wst:KeyType>
  </sp:RequestSecurityTokenTemplate>
</sp:IssuedToken>
```

The choice of using symmetric or asymmetric key tokens can depend on a variety of technical and business factors. For example, symmetric keys provide token processing speed and efficiency, whereas asymmetric keys may be needed to meet legal business requirements of non-repudiation of submitted tokens. The InfoCard model **recommends** that asymmetric key tokens be used whenever possible to enhance and safeguard user privacy for the reasons described below.

One of the ways the InfoCard model, and thus the identity selector implemented in Windows, strives to safeguard user privacy is by preventing the tracking of the user's online behavior by identity providers. In order to accomplish this goal the identity selector, by default, does not submit the relying party identity to the identity provider when requesting security tokens from it. However, this is not possible when using symmetric key tokens. The identity provider **must** encrypt the symmetric proof key to the relying party inside the issued security token and cover the encrypted key by the token signature to protect its integrity. This helps to scope the token specifically to the intended relying party. Placing a clear symmetric key in the issued security token, even if the token itself may be encrypted or conveyed over an encrypted channel, is not recommended. It poses a security threat wherein the relying party can repurpose and reuse the issued token acting as the token subject since it is in possession of the proof key. In order to encrypt the symmetric key specifically to the relying party, the identity of the relying party needs to be conveyed to the identity provider thus sacrificing user privacy.

Use of asymmetric key tokens does not suffer from the repurposing threat described above since the relying party is never in possession of the private key for the asymmetric proof key. Hence the use of asymmetric key tokens is recommended.

3.2.3. Freshness of issued tokens

Many service requester implementations may cache security tokens issued by identity providers until they expire for performance optimization. A relying party therefore needs a mechanism to indicate the maximum age of an issued security token (i.e. the maximum elapsed time since the instant the token was created) that it is willing to accept.

When a relying party wants to limit the maximum token age of a security token that it is willing to accept, the optional assertion element `ic:MaxTokenAge` defined in [InfoCard-Ref] may be used within an issued token policy assertion.

Following is an example of using this assertion within an issued token policy which indicates that the relying party will accept a token if it is less than 5 minutes (30,000 milliseconds) old since the time it was created.

Example:

```
<sp:IssuedToken ...>
  ...
  <wsp:Policy>
    <ic:MaxTokenAge>30000</ic:MaxTokenAge>
  </wsp:Policy>
</sp:IssuedToken>
```

3.2.4. Claims in issued tokens

A relying party can require that one or more claims about the subject be asserted by the identity provider in an issued token. If a required claim is absent in the token submitted to the relying party or is asserted with an empty value, the relying party can accept or reject that token at its own discretion.

When a relying party wants to convey its requirement for claims in issued tokens, the optional parameter element `wst:Claims` defined in [WS-Trust] may be used within an issued token policy assertion. The element `ic:Claim` defined in [InfoCard-Ref] can be used, within the `wst:Claims` element, to specify the individual claim types required and whether each specified claim type is *mandatory* or *optional*. When using the InfoCard model, the `wst:Claims` element should be qualified with the attribute `wst:Dialect` having the URI value of *http://schemas.microsoft.com/ws/2005/05/identity* that indicates how the specified claim elements are to be processed.

Following is an example of using this assertion within an issued token policy to require two claim types in the security token with one claim type being optional.

Example:

```
<sp:IssuedToken ...>
  ...
  <sp:RequestSecurityTokenTemplate>
    ...
    <wst:Claims
        wst:Dialect="http://schemas.microsoft.com/ws/2005/05/identity">
      <ic:Claim
        URI="http://.../ws/2005/05/identity/claims/givenname"/>
      <ic:Claim
        URI="http://.../ws/2005/05/identity/claims/surname"
        Optional="true" />
    </wst:Claims>
  </sp:RequestSecurityTokenTemplate>
  ...
</sp:IssuedToken>
```

The InfoCard Technical Reference [InfoCard-Ref] defines a standard set of claim types for typical personal information about users that is supported by the self-issued identity provider in the Windows InfoCard system. These claim types may be used by relying party Web services and identity providers in issued security tokens. Relying party Web services and identity providers may also use any other mutually understood claim types in the issued tokens that is relevant for them. Neither the InfoCard model nor the Windows InfoCard system places any constraint on the claim types that can be used in tokens.

3.2.5. Examples of endpoint and message policy

The following example shows a simple but fairly complete policy for a relying party service endpoint containing policy assertions defined in [WS-SecurityPolicy] and in [InfoCard-Ref].

Example:

Endpoint policy

```
<wsp:Policy
 wsu:Id="...#ServiceEndpoint_policy"
 xmlns="http://schemas.xmlsoap.org/ws/2004/08/policy" ...>

 <wsp:ExactlyOne>
  <wsp:All>
```

```
   <sp:SymmetricBinding>
    <wsp:Policy>
     <sp:ProtectionToken>
      <wsp:Policy>
       <sp:IssuedToken sp:IncludeToken=”.../IncludeToken/Once”>
        <sp:Issuer>
         <wsa:EndpointReference>
          <wsa:Address>
            http://schemas.microsoft.com/ws/2005/05/identity/issuer/self
          </wsa:Address>
         </wsa:EndpointReference>
        </sp:Issuer>
        <sp:RequestSecurityTokenTemplate>
         <wst:TokenType>urn:oasis:names:tc:SAML:1.0:assertion</wst:TokenType>
         <wst:KeyType>
           http://schemas.xmlsoap.org/ws/2004/04/trust/SharedKey
         </wst:KeyType>
         <wst:Claims
           wst:Dialect=”http://schemas.microsoft.com/ws/2005/05/identity”>
          <ic:Claim
            URI=”http://.../ws/2005/05/identity/claims/givenname”/>
          <ic:Claim
            URI=”http://.../ws/2005/05/identity/claims/surname”
            Optional=”true”/>
         </wst:Claims
        </sp:RequestSecurityTokenTemplate>
        <wsp:Policy>
         <ic:MaxTokenAge>65250000</ic:MaxTokenAge>
        </wsp:Policy>
       </sp:IssuedToken>
      </wsp:Policy>
     </sp:ProtectionToken>
     <sp:AlgorithmSuite>
      <wsp:Policy>
        [Algorithm Suite assertion]
      </wsp:Policy>
     </sp:AlgorithmSuite>
     <sp:Layout>
      <wsp:Policy>
       <sp:Strict />
      </wsp:Policy>
     </sp:Layout>
     <sp:IncludeTimestamp />
    </wsp:Policy>
   </sp:SymmetricBinding>
  </wsp:All>
 </wsp:ExactlyOne>
</wsp:Policy>
```

Per-message policy

```
<wsp:Policy
 wsu:Id="...#Service_message_policy"
 xmlns="http://schemas.xmlsoap.org/ws/2004/08/policy" ...>

 <wsp:ExactlyOne>
  <wsp:All>
    <sp:SignedParts>
```

```
     <sp:Body />
     <sp:Header
      Name="Action"
      Namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing" />
     <sp:Header
      Name="FaultTo"
      Namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing" />
     <sp:Header
      Name="From"
      Namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing" />
     <sp:Header
      Name="MessageID"
      Namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing" />
     <sp:Header
      Name="RelatesTo"
      Namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing" />
     <sp:Header
      Name="ReplyTo"
      Namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing" />
     <sp:Header
      Name="To"
      Namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing" />
     <sp:Header
      Name="RedirectTo"
      Namespace="http://schemas.microsoft.com/mb/2004/07/addressing" />
     <sp:Header
      Name="CoordinationContext"
      Namespace="http://schemas.xmlsoap.org/ws/2003/09/wscoor" />
    </sp:SignedParts>

    <sp:EncryptedParts>
     <sp:Body />
    </sp:EncryptedParts>
  </wsp:All>
 </wsp:ExactlyOne>
</wsp:Policy>
```

3.3. Retrieving Policy of Relying Party

The security policy of a relying party specifies its security token requirements and how messages should be secured on a communication path. Depending on the type of policy assertions, the security policy may apply to different types of target subjects such as service endpoints, individual messages, etc. The policies may be associated to the subject using mechanisms described in WS-PolicyAttachment.

Policy can be explicitly retrieved as specific metadata for an endpoint, or implicitly retrieved as part of other metadata of a service endpoint. A service requester can obtain the policy of the relying party either ahead of time or just-in-time, using the mechanisms specified in [WS-MetadataExchange], before sending any application request messages.

3.4. Submitting Security Tokens to Relying Party

The identity selector in the Windows InfoCard system is agnostic of specific token types, and can facilitate rendezvous between relying parties and identity providers based on arbitrary security token types and claim types.

Security tokens returned by the identity selector to the service requester application for submission to the relying party are encrypted to the key in the relying party endpoint identity that the user trusted and approved through a visual user interface. The service requester can submit the tokens by binding them to application messages using the security binding mechanisms described in [WS-SecurityPolicy]. Those binding mechanisms specify the security header layout for ordering of elements and signatures in messages.

3.4.1. Self-issued security tokens

When appropriate, the relying party may allow users to submit self-issued tokens to access a service. For example, when accessing a retail bookseller Web service to set up an account, the retail service may allow the user to self-assert her own name and address information in a self-issued security token. The InfoCard system in the Windows client platform includes the ability for users to produce self-issued security tokens. Appendix A describes the format of such self-issued tokens and how a relying party can evaluate and use them.

4. Identity Provider Interactions

As described earlier, an *InfoCard* represents a digital identity of the user issued by a corresponding identity provider. Multiple digital identities for the user from the same identity provider would be represented by different InfoCards. Users may have a collection of InfoCards representing the various digital identities they have from identity providers. An InfoCard itself is ***not*** the security token that is used to carry identity claims in Web service protocols, rather it is an artifact that represents the token issuance relationship that the user has with the identity provider. An actual security token with specific claims can be requested from the identity provider when needed based on the InfoCard.

This section describes the general framework used by identity providers to issue digital identities to users in the form of InfoCards, and for the identity selector in Windows to request security tokens from identity providers based on those InfoCards. An identity provider runs one or more instances of security token services as shown in Figure 1 to process requests for security tokens based on InfoCards issued to users. For brevity, the identity provider STS will be referred to as “IP/STS” in the remainder of this section.

At a high-level, an InfoCard carries the identity provider’s issuance policy for security tokens including the types of tokens it can issue, the claim types it is authoritative for, and the credential to use for authentication when requesting security tokens. The InfoCard therefore contains sufficient information about the specific capabilities of the identity provider, with regards to the digital identity of the user that it represents, to allow the identity selector on a service requester to match it with a relying party’s token requirements. Once a matching InfoCard that can satisfy a relying party’s token policy is selected by the user, the identity selector requests and obtains the appropriate security token from the IP/STS using the mechanisms described in [WS-Trust].

4.1. InfoCard

Users may obtain an artifact with a unique identifier from an identity provider, called an *InfoCard*, using which users can visualize their digital relationship with the identity provider in user interfaces and request security tokens with claims from the identity provider. InfoCards serve the very important purpose of transforming something abstract as digital identities into something concrete and tangible for users that they can work with in their everyday digital interactions. Furthermore, being concrete artifacts, the InfoCards representing a user’s digital identities are portable and can be carried around by the user and employed from any computer or device through which Web services are accessed.

4.1.1. InfoCard format

InfoCards are concretely represented as XML documents that can be issued by identity providers and stored by users on any storage device of their choice. An InfoCard does not inherently contain any security sensitive data as it simply represents the security token issuance relationship of the user with an identity provider. The actual request and issuance of the corresponding security tokens will be authenticated and secure.

The XML schema for an InfoCard is described in [InfoCard-Ref] and is represented by the `ic:InfoCard` element. The `xml:lang` attribute can be used, either at the root element level or at any of the child elements within it, to specify the language in which the content of the localizable child elements in the InfoCard have been localized.

The following example illustrates an InfoCard issued by the “XYZ Authority” that supports the SAML token type, two claim types, requires the relying party identity to be conveyed in token requests, and requires authentication based on username/password when requesting tokens.

Example:

```
<InfoCard
    xmlns="http://schemas.microsoft.com/ws/2005/05/identity"
    xmlns:wsa=”http://schemas.xmlsoap.org/ws/2004/08/addressing”
    xmlns:wsp=”http://schemas.xmlsoap.org/ws/2002/12/policy”
    xml:lang=”en-us”>
  <InfoCardReference>
    <CardId>http://xyz.com/CardId/d795621fa01d454285f9</CardId>
  </InfoCardReference>
  <CardName>XYZ membership card</CardName>
  <CardImage MimeType=”image/gif”> ... </CardImage>
  <IssuerName>XYZ Authority</IssuerName>
  <TimeIssued>2003-08-24T00:30:05Z</TimeIssued>
  <TokenServiceReference>
    <TokenService>
      <wsa:EndpointReference>
        <wsa:Address>http://xyz.org/sts</wsa:Address>
        <wsid:Identity>
          <ds:KeyInfo>
            <ds:X509Data>
              <ds:X509Certificate>...</ds:X509Certificate>
            </ds:X509Data>
          </ds:KeyInfo>
        </wsid:Identity>
      </wsa:EndpointReference>
      <UserNamePasswordAuthenticate>
          <Username>Zoe</Username>
      </UserNamePasswordAuthenticate>
    </TokenService>
  </TokenServiceReference>
  <ic:InfoCardPolicy>
    <SupportedTokenTypes>
      <TokenType URI=”urn:oasis:names:tc:SAML:1.0:assertion”/>
    </SupportedTokenTypes>
    <SupportedClaims>
      <SupportedClaim URI=”http://.../ws/2005/05/identity/claims/givenname”>
        <DisplayTag>Given Name</DisplayTag>
      </SupportedClaim>
      <SupportedClaim URI=”http://.../ws/2005/05/identity/claims/surname”>
```

```
        <DisplayTag>Last Name</DisplayTag>
      </SupportedClaim>
    </SupportedClaims>
    <RequireAppliesTo />
  </ic:InfoCardPolicy>
</InfoCard>
```

4.1.2. Issuing InfoCards

An identity provider may issue InfoCards to its users using any convenient out-of-band mechanism mutually suited to the user and the identity provider. For example, a user may log on to a Web site provided by the identity provider and download the InfoCard over the HTTP connection. Alternatively, an identity provider may send a user's InfoCard through email to the user's email address on file. Remember that the InfoCard is *not* the security token; it represents the token issuance relationship that the user has with the identity provider.

In order to provide some assurance to the user that an InfoCard was indeed issued by the identity provider expected by the user, the InfoCard should be carried inside a digitally signed envelope (i.e. enveloping signature) signed by the identity provider. The digitally signed envelope is the moral equivalent of a signed message, which provides a signed container, in which the application payload gets delivered to a recipient. In this case, the InfoCard itself is included as the element content of the `ds:Object` element. The signature on the digitally signed envelope makes no implicit statement about the content of the InfoCard itself that is carried inside it except providing data origin authentication.

We recommend that X509v3 certificates for the identity provider organization (as opposed to SSL server certificates) with logotypes for the issuer organization and subject organization [RFC 3709] be used for this purpose. Such certificates can be used by the user interface on the user's system to visually identify the identity provider organization before saving the InfoCard for later use.

In addition, an X509v3 certificate for the identity provider organization should also be included as the endpoint identity in each IP/STS endpoint reference specified in the InfoCard using the `wsid:Identity` extension element as described in [InfoCard-Ref]. If message security is used, this endpoint identity is used by the identity selector to establish a secure message channel with the IP/STS backing the InfoCard when requesting security tokens.

The specific details of the XML digital signature [XMLDSIG] profile that should be used to sign the envelope carrying the InfoCard using an organizational X509v3 certificate is described in [InfoCard-Ref]. The following example shows the general form of an InfoCard within an enveloping signature using that prescribed format.

Example:

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod
      Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
    <SignatureMethod
      Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
    <Reference URI="#_Object_InfoCard">
      <Transforms>
        <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
      <DigestValue> ... </DigestValue>
    </Reference>
```

```
  </SignedInfo>
  <SignatureValue> ... </SignatureValue>
  <KeyInfo>
    <X509Data>
      <X509Certificate> ... </X509Certificate>
    </X509Data>
  </KeyInfo>
  <Object Id="_Object_InfoCard">
    <ic:InfoCard
        xmlns:ic="http://schemas.microsoft.com/ws/2005/05/identity"
        xml:lang="en-us">
      [Actual InfoCard content]
    </ic:InfoCard>
  </Object>
</Signature>
```

Once the user's system has verified the integrity of the InfoCard and visually identified the identity provider organization to the user on a suitable user interface, the InfoCard represented by the `ic:InfoCard` element can be extracted from the enveloping signature for storage.

4.2. Token Issuance Policy of Identity Provider

This section describes the various policy assertion elements that can be used in an InfoCard issued by an identity provider to represent its token issuance policy (see the InfoCard schema defined in [InfoCard-Ref]). One or more of these token issuance policy assertions can appear as children of the `ic:InfoCardPolicy` element inside the InfoCard.

4.2.1. Supported token types

The token issuance policy for an InfoCard allows an identity provider to express the list of security token types that it can issue. The policy element `ic:SupportedTokenTypes` defined in [InfoCard-Ref] may be used within an InfoCard policy for this purpose.

The following example illustrates a supported tokens policy of an identity provider that can issue both SAML 1.1 and SAML 2.0 tokens.

Example:

```
<ic:InfoCardPolicy>
  <ic:SupportedTokenTypes>
    <ic:TokenType URI="urn:oasis:names:tc:SAML:1.0:assertion" />
    <ic:TokenType URI="urn:oasis:names:tc:SAML:2.0:assertion" />
  </ic:SupportedTokenTypes>
</ic:InfoCardPolicy>
```

4.2.2. Supported claim types

The token issuance policy for an InfoCard allows an identity provider to express the list of claim types it can assert in security tokens it issues. The policy element `ic:SupportedClaims` defined in [InfoCard-Ref] may be used within an InfoCard policy for this purpose.

The following example illustrates a supported claim types policy of an identity provider.

Example:

```
<ic:InfoCardPolicy>
  <ic:SupportedClaims>
    <ic:SupportedClaim URI="http://.../ws/2005/05/identity/claims/givenname">
```

```
      <ic:DisplayTag>Given Name</ic:DisplayTag>
      <ic:Description>First name of a person</ic:Description>
    </ic:SupportedClaim>
    <ic:SupportedClaim URI="http://.../ws/2005/05/identity/claims/surname">
      <ic:DisplayTag>Last Name</ic:DisplayTag>
      <ic:Description>Last name or family name of a person</ic:Description>
    </ic:SupportedClaim>
  </ic:SupportedClaims>
</ic:InfoCardPolicy>
```

4.2.3. Require relying party identity

One of the ways by which the InfoCard model, and thus the identity selector implemented in Windows, safeguards user privacy is by preventing the tracking of the user's online behavior by identity providers. In order to accomplish this goal the identity selector, by default, does not submit the relying party identity to the identity provider when requesting security tokens. However, the identity provider can override that behavior through policy if there are justifiable reasons to do so (e.g. audit requirements for compliance).

The token issuance policy for an InfoCard allows an identity provider to assert that the identity of the relying party be conveyed in the `wsp:AppliesTo` element of the token request message to the IP/STS supporting that InfoCard. The policy assertion element `ic:RequireAppliesTo` defined in [InfoCard-Ref] may be used within an InfoCard policy for this purpose.

Following is an example of this assertion used in the issuance policy within an InfoCard.

Example:

```
<ic:InfoCardPolicy>
  <ic:RequireAppliesTo />
</ic:InfoCardPolicy>
```

4.3. Security Token Request to Identity Provider

When the user selects an InfoCard on a service requester machine for submission to a relying party in a given interaction context, the identity selector on that system obtains a suitable security token from the IP/STS for that InfoCard. Security tokens are requested using the issuance mechanism described in [WS-Trust]. Specifically, tokens are requested by submitting a RequestSecuriyToken (RST) message to the IP/STS. This section describes the specific constraints and/or extensions to the token request messages sent to the IP/STS as described in [InfoCard-Ref].

4.3.1. Supplying InfoCard reference

Each InfoCard has a unique identifier and version, given by the `ic:InfoCardReference` element in an InfoCard, by which it can be referenced when requesting security tokens from the IP/STS for the InfoCard. When requesting a security token from the IP/STS, the InfoCard reference is included in the body of the "request security token" (RST) message as a top-level element information item as described in [InfoCard-Ref].

Following is an example of the InfoCard reference being included in a RST message.

Example:

```
<wst:RequestSecurityToken>
  ...
  <ic:InfoCardReference>
    <ic:CardId>http://xyz.com/CardId/d795621fa01d454285f9</ic:CardId>
    <ic:CardVersion>1</ic:CardVersion>
```

```
  </ic:InfoCardReference>
  ...
</wst:RequestSecurityToken>
```

The InfoCard reference is only meaningful to the IP/STS. It may use the InfoCard reference information to determine if the corresponding InfoCard is stale or out-of-date for whatever reason. The IP/STS may fault with `ic:InfoCardRefreshRequired` defined in [InfoCard-Ref] to signal to the service requester that the InfoCard needs to be refreshed.

4.3.2. Requested claims and other parameters from relying party

A relying party may request a specific set of claims along with other request parameters that must be communicated to the IP/STS. These request parameters are expressed in the token requirement policy of the relying party using the `sp:RequestSecurityTokenTemplate` element within the `sp:IssuedToken` policy assertion (see Section 3.2). The content of this element (i.e. all of its [children] elements) are copied directly into the RST request message sent to the IP/STS.

For example, let us suppose the relying party asks for an issued token in its security policy as follows.

```
<sp:IssuedToken>
  <sp:RequestSecurityTokenTemplate>
    <wst:KeyType>
      http://schemas.xmlsoap.org/ws/2004/04/trust/PublicKey
    </wst:KeyType>
    <wst:Claims
        wst:Dialect="http://schemas.microsoft.com/ws/2005/05/identity">
      <ic:Claim
        URI="http://.../ws/2005/05/identity/claims/givenname"/>
      <ic:Claim
        URI="http://.../ws/2005/05/identity/claims/surname"
        Optional="true" />
    </wst:Claims>
  </sp:RequestSecurityTokenTemplate>
</sp:IssuedToken>
```

The identity selector on the service requester copies the entire content of the element `sp:RequestSecurityTokenTemplate` into the RST request message when requesting the security token as follows.

Example:

```
<wst:RequestSecurityToken>
  <wst:KeyType>
    http://schemas.xmlsoap.org/ws/2004/04/trust/PublicKey
  </wst:KeyType>
  <wst:Claims
      wst:Dialect="http://schemas.microsoft.com/ws/2005/05/identity">
    <ic:Claim
      URI="http://.../ws/2005/05/identity/claims/givenname"/>
    <ic:Claim
      URI="http://.../ws/2005/05/identity/claims/surname"
      Optional="true" />
  </wst:Claims>
  ...
</wst:RequestSecurityToken>
```

4.3.3. Identifying relying party to identity provider

The identity selector, by default, does not submit the relying party identity to the identity provider when requesting security tokens to protect user privacy. Instead, a hashed function of the relying party identity and optional user-supplied entropy is submitted in the security token request as an opaque reference to the relying party. This opaque handle can be used by the identity provider to reproducibly synthesize any pair-wise claims or data specific to that relying party such as pair-wise subject identifiers or pseudonyms for users. However, the actual identity of the relying party is communicated to the IP/STS if either of the following is true:

- The relying party requires a *symmetric key* token.
- The token issuance policy of the identity provider includes the `ic:RequireAppliesTo` policy assertion element (within the InfoCard).

The `ic:OpaqueEndpoint` element defined in [InfoCard-Ref] is used to send an opaque reference for a relying party identity to an IP/STS.

An opaque reference for the relying party identity is submitted to the IP/STS by including a `wsp:AppliesTo` element containing the opaque handle for the relying party endpoint in the RST request message as shown in the following example.

Example:

```
<wst:RequestSecurityToken>
  <wsp:AppliesTo>
    <ic:OpaqueEndpoint>
      MIIEZzCCA9CgAwIBAgIQEmtJZc0==
    </ic:OpaqueEndpoint>
  </wsp:AppliesTo>
  ...
</wst:RequestSecurityToken>
```

The actual relying party identity is submitted to the IP/STS by including a `wsp:AppliesTo` element containing the endpoint reference of the relying party in the RST request message as shown in the following example.

Example:

```
<wst:RequestSecurityToken>
  <wsp:AppliesTo>
    <wsa:EndpointReference>
      <wsa:Address>http://ip.fabrikam.com/STS</wsa:Address>
      ...
    </wsa:EndpointReference>
  </wsp:AppliesTo>
  ...
</wst:RequestSecurityToken>
```

4.3.4. Requesting display token

The identity selector on a service requester is agnostic of the specific type of security tokens that may be requested by a relying party and issued by an identity provider. In fact, the token returned by an IP/STS may be completely opaque to the service requester. However, it is sometimes useful for the service requester to request and obtain an informational token along with the actual security token from the identity provider that could be displayed to the user in a user interface to allow informed user consent and release. The informational token, called a *display token* in the InfoCard model, essentially contains a textual representation of the claims carried in the accompanying security token.

When requesting a display token to be returned along with the security token issued by the IP/STS, the optional ic:RequestDisplayToken element defined in [InfoCard-Ref] is included in the RST request message from the service requester.

A security token request that also includes a request for a display token in the US English language is shown in the following example.

Example:

```
<wst:RequestSecurityToken>
  <ic:RequestDisplayToken xml:lang="en-us" />
</wst:RequestSecurityToken>
```

The format of the display token returned by the IP/STS in a response RSTR message is described later in Section 4.4.1.

4.3.5. Requesting proof key

The optional wst:KeyType element in the RST request indicates the type of key desired in the security token. The default bahavior for an identity provider is to issue a symmetric key token if no explicit key type is specified in the token request (as specified in [WS-Trust]). The identity selector, when requesting a security token from an identity provider, communicates the requested key type exactly as specified by the relying party in its issued token policy.

When a *symmetric key* token is requested from the IP/STS, the identity selector on the service requester does not supply any key material for the proof key in the request. So the IP/STS must generate the symmetric key for use in the token and return it in the response.

When an *asymmetric key* token is requested from the IP/STS the the identity selector on the service requester generates an ephemeral **1024-bit RSA key pair** for use as the proof key and submits the public key to the IP/STS by augmenting the RST message as follows:

- It includes a wst:KeyType element with the following URI value in the RST message to indicate that an asymmetric key based token is requested.

  http://schemas.xmlsoap.org/ws/2004/04/security/trust/PublicKey

- It includes a wst:UseKey element in the RST message to indicate the public key to be used in the returned security token.
- It includes a supporting signature to prove ownership of the corresponding private key. The ds:KeyInfo element within the signature includes the actual public key in the form of a ds:RSAKeyValue child element of a ds:KeyValue element.
- The supporting signature is placed in the security header of the RST message where the signature for an endorsing supporting token would be placed as per the security header layout described in [WS-SecurityPolicy].

Following is a sample RST request fragment that illustrates an asymmetric key based token request containing the public key and proof of ownership of the corresponding private key.

Example:

```
<s:Envelope ... >
  <s:Header>
    ...
    <wsse:Security>
      ...
      <ds:Signature Id="_proofSignature">
        <!-- signature proving possession of requested public key -->
```

```
        <!-- KeyInfo in signature contains the public key or reference -->
        <KeyInfo>
          <KeyValue>
            <RSAKeyValue>
              <Modulus>...</Modulus>
              <Exponent>...</Exponent>
            </RSAKeyValue>
          </KeyValue>
        </KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </s:Header>
  <s:Body wsu:Id="req">
    <wst:RequestSecurityToken>
      ...
      <wst:KeyType>
        http://schemas.xmlsoap.org/ws/2004/04/security/trust/PublicKey
      </wst:KeyType>
      <wst:UseKey Sig="#_proofSignature" />
    </wst:RequestSecurityToken>
  </s:Body>
</s:Envelope>
```

4.4. Security Token Response from Identity Provider

This section describes the token issuance behavior of an identity provider in the InfoCard model in response to the request characteristics described earlier.

4.4.1. Returning display token

If a service requester includes a request in the RST message for a display token (see Section 4.3.4) corresponding to the returned security token, then the IP/STS may optionally create and return the display token in the RSTR response message to the requester. Further, if a language was specified in the request using the `xml:lang` attribute then the returned display token should be localized in that language.

To return a display token, the IP/STS can use the optional `ic:RequestedDisplayToken` element defined in [InfoCard-Ref] in the RSTR response message. The `xml:lang` attribute is used to specify the language in which the returned display token is localized.

The following example illustrates a returned display token localized in “US English” corresponding to a requested security token carrying two claims.

Example:

```
<ic:RequestedDisplayToken>
  <ic:DisplayToken xml:lang=”en-us”>
    <ic:DisplayClaim URI=”http://.../ws/2005/05/identity/claims/givenname”>
      <ic:DisplayTag>Given Name</ic:DisplayTag>
      <ic:DisplayValue>John</ic:DisplayValue>
    </ic:DisplayClaim>
    <ic:DisplayClaim URI=”http://.../ws/2005/05/identity/claims/surname”>
      <ic:DisplayTag>Last Name</ic:DisplayTag>
      <ic:DisplayValue>Doe</ic:DisplayValue>
    </ic:DisplayClaim>
  <ic:DisplayToken>
</ic:RequestedDisplayToken>
```

4.4.2. Returning proof key

The IP/STS can return a proof-of-possession key along with the issued token by including a `wst:RequestedProofToken` element in the RSTR response. This section describes when this element should be included in the response message and what its content should be. Note that the token response is always carried over a confidential channel where either an encrypted transport is used (transport security) or the body of the response SOAP message is encrypted to the requester (messsage security).

When a *symmetric key* token is requested, the requester InfoCard system does not supply any key material. So the IP/STS **must** generate the symmetric key for use in the token. The proof key is returned in the response by including a `wst:RequestedProofToken` element with a `wst:BinarySecret` child element carrying the base64 encoded symmetric key. Alternatively, the symmetric proof key can be encrypted to the requester's authentication key used in the request and returned as an `xenc:EncryptedKey` child of the `wst:RequestedProofToken` element.

When an *asymmetric key* token is requested, the requester InfoCard system supplies a public key, along with proof of ownership of the private key, to be used in the issued token. If the IP/STS accepts the supplied public key for use in the issued token, then a proof token should not be returned. In other words, the proof key is implied and the response should not include a `wst:RequestedProofToken` element.

However, the IP/STS can override the asymmetric proof key supplied by the requester InfoCard system and specify a public key token of its choice (for which the requester already has the private key) as the proof token. In that case, a `wst:RequestedProofToken` element must be returned in the response containing a `ds:KeyInfo` child element that provides the selection criteria for the specific proof token. The Windows InfoCard system can only accept an X509v3 certificate (either software or hardware token based) as the overriding proof token. The X509v3 certificate to be used as proof token can be specified in the way given below:

```
<!-- Select certificate based on certificate thumbprint -->
<wst:RequestedProofToken>
  <ds:KeyInfo>>
    <wsse:SecurityTokenReference>
      <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/2004/xx/
oasis-2004xx-wss-soap-message-security-1.1#ThumbprintSHA1">
        ...
      </wsse:KeyIdentifier>
    </wsse:SecurityTokenReference>
  </ds:KeyInfo>
</wst:RequestedProofToken>
```

To enable the requester InfoCard system to locate the right X509 certificate for use, the SHA1 hash of the entire certificate content should be included as a "thumbprint" in the requested proof token to allow the exact certificate to be located. The thumbprint search value can be used with the appropriate platform-specific APIs (e.g. CAPI2 on Windows) to locate the right certificate.

The following table summarizes the proof key generation responsibilities in the InfoCard model:

Type of proof key in token	Supplied by requester InfoCard system	Supplied by IP/STS
Symmetric key	nothing	Proof key generated by IP/STS;

		proof key encrypted to relying party inside security token; proof key explicitly returned in RSTR response (optionally encrypted to requester authentication token)
Asymmetric key	Requester InfoCard system generates temporary RSA key pair and submits public key in RST request (along with proof of possession of private key) for use as proof key in issued token	If proof key is accepted by IP/STS, then it is included in issued token and no proof key is explicitly returned in RSTR; If proof key is overridden by IP/STS, then RSTR response includes reference for proof key token for which requester already has the private key

5. Identity Provider Message Exchanges

The InfoCard schema includes the element content necessary for an identity provider to express what credential the user must use in order to authenticate to the IP/STS when requesting tokens. For each supported credential type, this section describes in detail:

- the schema used to express the credential selection criteria,
- the suggested security binding that the requester and the IP/STS should use, and
- the format and content of request and response messages between the identity selector and the IP/STS.

Note that the identity selector will retrieve the actual security policy from the IP/STS using WS-MetadataExchange to determine the security binding to use. If the implied credential type in the retrieved security policy does not match the corresponding credential selector specified in the InfoCard, then the identity selector will fail without sending the token request.

The identity selector submits token requests to the IP/STS by using the security binding mechanisms described in [WS-SecurityPolicy]. Those binding mechanisms specify the security header layout for ordering of elements and signatures in messages.

The XML signature [XMLDSIG] profile that will be used for signatures placed in request and response messages for message authentication and integrity is governed by the value of the `sp:AlgorithmSuite` assertion element in the security policy of the IP/STS as described in [WS-SecurityPolicy].

Similarly, the XML encryption [XMLENC] profile that will be used for encrypted keys and encrypted element content placed in request and response messages for confidentiality is governed by the value of the `sp:AlgorithmSuite` assertion element in the security policy of the IP/STS as described in [WS-SecurityPolicy].

5.1. Retrieving Security Policy of Identity Provider

When an InfoCard is selected by the user for use with a relying party, the identity selector on the service requester prepares to request a security token from the corresponding identity provider by first fetching its security policy. The IP/STS endpoint specified in an InfoCard issued by the identity provider must have a metadata exchange endpoint that responds to policy requests from a service requester using [WS-MetadataExchange]. The security policy of the IP/STS indicates endpoint behavior over a token request/response

sequence, and specifies its security binding for how messages should be secured on the channel. A policy containing the security policy assertions should be attached to the **wsdl:binding** in the WSDL for the IP/STS. The identity selector on a service requester will obtain the policy of the IP/STS using the mechanisms specified in WS-MetadataExchange.

The security policy assertions for the endpoint are specific to the authentication mechanism required by the IP/STS and are described in the sections that follow. The suggested message integrity and confidentiality policy assertions for the token request/response messages are the same for all authentication mechanisms and are as follows.

Per-message policy:

```
<wsp:Policy>
 <wsp:ExactlyOne>
  <wsp:All>
    <sp:SignedParts>
     <sp:Body />
     <sp:Header
      Name="Action"
      Namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing" />
     <sp:Header
      Name="FaultTo"
      Namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing" />
     <sp:Header
      Name="From"
      Namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing" />
     <sp:Header
      Name="MessageID"
      Namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing" />
     <sp:Header
      Name="RelatesTo"
      Namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing" />
     <sp:Header
      Name="ReplyTo"
      Namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing" />
     <sp:Header
      Name="To"
      Namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing" />
     <sp:Header
      Name="RedirectTo"
      Namespace="http://schemas.microsoft.com/mb/2004/07/addressing" />
    </sp:SignedParts>

    <sp:EncryptedParts>
     <sp:Body />
    </sp:EncryptedParts>
  </wsp:All>
 </wsp:ExactlyOne>
</wsp:Policy>
```

5.1.1. Message exchange

If the identity selector does not already have the security policy of the IP/STS, it performs get metadata operation to fetch policy. It is an unsecured request/response exchange using the WS-MetadataExchange protocol.

Requester to IP/STS metadata request message:

```
<S:Envelope ...>
```

```
   <S:Header>
    <wsa:Action S:mustUnderstand="1">
      http://schemas.xmlsoap.org/ws/2004/08/mex/GetMetadata/Request
    </wsa:Action>
    <wsa:MessageID>
      uuid:ab9e1c77-0cea-4f2f-a586-78c15536137d
    </wsa:MessageID>
    <wsa:To S:mustUnderstand="1">
      http://contoso.com/IP/STS/Mex
    </wsa:To>
    <wsa:ReplyTo>
      <wsa:Address>
        http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
      </wsa:Address>
    </wsa:ReplyTo>
  </S:Header>
  <S:Body>
    <GetMetadata xmlns="http://schemas.xmlsoap.org/ws/2004/08/mex" />
  </S:Body>
</S:Envelope>
```

Note the following in the metadata request message:

- The request is unsecured.
- The GetMetadata element does not specify any specific metadata dialect causing all metadata to be returned by the metadata exchange endpoint of the IP/STS.

IP/STS to Requester metadata response message:

```
<S:Envelope ...>
 <S:Header>
  <wsa:Action S:mustUnderstand="1">
    http://schemas.xmlsoap.org/ws/2004/08/mex/GetMetadata/Response
  </wsa:Action>
  <wsa:RelatesTo>
    uuid:ab9e1c77-0cea-4f2f-a586-78c15536137d
  </wsa:RelatesTo>
  <wsa:To S:mustUnderstand="1">
    http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
  </wsa:To>
 </S:Header>

 <S:Body>
  <Metadata xmlns="http://schemas.xmlsoap.org/ws/2004/08/mex">
   <MetadataSection Dialect="http://schemas.xmlsoap.org/wsdl/"
     Identifier="http://contoso.com/IP/STS/Mex/mexRoot">
    <wsdl:definitions
      targetNamespace="http://contoso.com/IP/STS/Mex/mexRoot" ...>
     <wsdl:service name="Service">
      ...
     </wsdl:service>
    </wsdl:definitions>
   </MetadataSection>

   <MetadataSection Dialect="http://schemas.xmlsoap.org/wsdl/"
     Identifier="http://contoso.com/IP/STS/Mex">
    <wsdl:definitions
      targetNamespace="http://schemas.xmlsoap.org/ws/2004/04/trust" ...>
```

```
     <wsdl:types>
      <xs:schema>
       <xs:import namespace="http://schemas.xmlsoap.org/ws/2004/04/trust"
         schemaLocation="ws-trust.xsd"/>
      </xs:schema>
     </wsdl:types>
     ...
    </wsdl:definitions>
   </MetadataSection>

   <MetadataSection Dialect="http://schemas.xmlsoap.org/wsdl/"
     Identifier="http://contoso.com/IP/STS/Mex/bindings">
    <wsdl:definitions
      targetNamespace="http://contoso.com/IP/STS/Mex/bindings" ...>
     <wsdl:binding>
      ...
     </wsdl:binding>
     <wsp:Policy wsu:Id="_Service_Binding_Policy" ...>
      <wsp:ExactlyOne>
       <wsp:All> [security binding assertions] </wsp:All>
      </wsp:ExactlyOne>
     </wsp:Policy>
     <wsp:Policy wsu:Id="_Input_Message_Policy" ...>
      <wsp:ExactlyOne>
       <wsp:All>[message integrity and confidentiality assertions]</wsp:All>
      </wsp:ExactlyOne>
     </wsp:Policy>
     <wsp:Policy wsu:Id="_Output_Message_Policy" ...>
      <wsp:ExactlyOne>
       <wsp:All>[message integrity and confidentiality assertions]</wsp:All>
      </wsp:ExactlyOne>
     </wsp:Policy>
     ...
    </wsdl:definitions>
   </MetadataSection>
  </Metadata>
 </S:Body>
</S:Envelope>
```

Note the following in the metadata response message:

- Since all metadata was requested, several metadata sections are returned in the response each containing a specific type of metadata – namely, service contract definitions, bindings including policy and where each policy is attached (*i.e.* to what target subject does each policy apply).

5.2. Authenticating with Username and Password

The identity provider requires that the service requester submit a *username* and corresponding *password* as the credential to authenticate to the IP/STS when requesting security tokens.

5.2.1. Credential format

The credential selector format for username/password defined in [InfoCard-Ref] has the following form:

```
<ic:UserNamePasswordAuthenticate>
```

```
  <ic:Username>zoe</ic:Username>
</ic:UserNamePasswordAuthenticate>
```

For user convenience, the “username” can be optionally included in the ic:Username element of the credential description in the InfoCard as shown in the example above. The user will be asked to supply the “password” when the InfoCard is selected for use.

5.2.2. Security binding

The “transport” security binding as described by [WS-SecurityPolicy] should be used for this authentication method for the request and response messages.

An example of the suggested policy assertions for this security binding is as follows (Note that the requester identity selector will retrieve the actual security policy from the IP/STS using WS-MetadataExchange to determine the security binding to use).

Endpoint policy:

```
<wsp:Policy>
  <sp:TransportBinding>
    <wsp:Policy>
      <sp:TransportToken>
        <wsp:Policy>
          <sp:HttpsToken />
        </wsp:Policy>
      </sp:TransportToken>
      <sp:AlgorithmSuite>
        <wsp:Policy>
          [Algorithm Suite assertion]
        </wsp:Policy>
      </sp:AlgorithmSuite>
      <sp:Layout>
        <wsp:Policy>
          <sp:Strict />
        </wsp:Policy>
      </sp:Layout>
      <sp:IncludeTimestamp />
      <sp:SupportingTokens>
        <wsp:Policy>
          <sp:UsernameToken sp:IncludeToken=”.../IncludeToken/Once” />
        </wsp:Policy>
      </sp:SupportingTokens>
    </wsp:Policy>
  </sp:TransportBinding>
</wsp:Policy>
```

5.2.3. Message exchange

For this security binding, message protection and security correlation for the request and response legs of the message exchange is provided by a secure HTTPS transport as described in [WS-SecurityPolicy]. There is no message level encryption required. A request and response message based on the security policy specified in the previous section is illustrated below.

Requester to IP/STS request message:

```
<S:Envelope ...>
   <S:Header>
    <wsa:Action wsu:Id="_1">
      http://schemas.xmlsoap.org/ws/2004/04/security/trust/RST/Issue
```

```
    </wsa:Action>
    <wsa:MessageID wsu:Id="_2">
      uuid:eb9e1c77-0cea-4f2f-a586-78c15536137c
    </wsa:MessageID>
    <wsa:To wsu:Id="_3">
      http://contoso.com/IP/STS
    </wsa:To>
    <wsa:ReplyTo wsu:Id="_4">
      <wsa:Address>
        http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
      </wsa:Address>
    </wsa:ReplyTo>
    <wsse:Security S:mustUnderstand="1">
      <wsu:Timestamp wsu:Id="_6">
        <wsu:Created>2004-10-18T09:02:00Z</wsu:Created>
        <wsu:Expires>2004-10-18T09:12:00Z</wsu:Expires>
      </wsu:Timestamp>
      <!-- Username w/ cleartext password as authentication token -->
      <wsse:UsernameToken wsu:Id="_6">
        <wsse:Username>Zoe</wsse:Username>
        <wsse:Password
            Type="http:// http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-username-token-profile-1.0#PasswordText">
          ILoveDogs
        </wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </S:Header>
  <S:Body wsu:Id=”_10”>
    <wst:RequestSecurityToken>
      <wst:TokenType>
          urn:oasis:names:tc:SAML:1.0:assertion
      </wst:TokenType>
      <wst:RequestType>
          http://schemas.xmlsoap.org/ws/2004/04/security/trust/Issue
      </wst:RequestType>
      <ic:InfoCardReference>
        <ic:CardId>http://xyz.com/CardId/d795621fa01d454285f9</ic:CardId>
      </ic:InfoCardReference>
      <wst:Claims
          wst:Dialect=”http://schemas.microsoft.com/ws/2005/05/identity”>
        <ic:Claim
          URI=”http://.../ws/2005/05/identity/claims/givenname”/>
        <ic:Claim
          URI=”http://.../ws/2005/05/identity/claims/surname”/>
      </wst:Claims>
      <ic:RequestDisplayToken xml:lang="en-us" />
    </wst:RequestSecurityToken>
  </S:Body>
</S:Envelope>
```

Note the following in the request message:

- The username token conveys the password in clear text.
- The InfoCard reference (CardId) is included in the RST message element.
- A display token localized in “en-us” is requested.

IP/STS to Requester response message:

```
<S:Envelope ...>
   <S:Header>
    <wsa:Action wsu:Id="_1">
      http://schemas.xmlsoap.org/ws/2004/04/security/trust/RSTR/Issue
    </wsa:Action>
    <wsa:RelatesTo wsu:Id="_2">
      uuid:eb9e1c77-0cea-4f2f-a586-78c15536137c
    </wsa:RelatesTo>
    <wsa:To wsu:Id="_3">
      http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
    </wsa:To>
    <wsse:Security S:mustUnderstand="1">
      <wsu:Timestamp wsu:Id="_6">
        <wsu:Created>2004-10-18T09:02:00Z</wsu:Created>
        <wsu:Expires>2004-10-18T09:12:00Z</wsu:Expires>
      </wsu:Timestamp>
    </wsse:Security>
  </S:Header>
  <S:Body wsu:Id=”_10”>
    <wst:RequestSecurityTokenResponse>
      <wst:TokenType>
          urn:oasis:names:tc:SAML:1.0:assertion
      </wst:TokenType>
      <wst:Lifetime>
        <wsu:Created>2004-10-18T09:02:00Z</wsu:Created>
        <wsu:Expires>2004-10-18T09:12:00Z</wsu:Expires>
      </wst:Lifetime>
      <wst:RequestedSecurityToken>
        <saml:Assertion xmlns="urn:oasis:names:tc:SAML:1.1:assertion" ...>
          ...
        </saml:Assertion>
      </wst:RequestedSecurityToken>
      <wst:RequestedAttachedReference>
        <wsse:SecurityTokenReference>
          <wsse:KeyIdentifier ValueType="...#SAMLAssertionID">
             _SAMLAssertionID
          </wsse:KeyIdentifier>
        </wsse:SecurityTokenReference>
      </wst:RequestedAttachedReference>
      <wst:RequestedProofToken>
        <wst:BinarySecret>
          +amQ87tQFs67gkiet==
        </wst:BinarySecret>
      </wst:RequestedProofToken>
      <ic:RequestedDisplayToken>
       <ic:DisplayToken xml:lang="en-us">
        <ic:DisplayClaim
            URI=”http://.../ws/2005/05/identity/claims/givenname”>
          <ic:DisplayTag>Given Name</ic:DisplayTag>
          <ic:DisplayValue>John</ic:DisplayValue>
        </ic:DisplayClaim>
        <ic:DisplayClaim
            URI=”http://.../ws/2005/05/identity/claims/surname”>
          <ic:DisplayTag>Last Name</ic:DisplayTag>
          <ic:DisplayValue>Doe</ic:DisplayValue>
```

```
        </ic:DisplayClaim>
       </ic:DisplayToken>
      </ic:RequestedDisplayToken>
    </wst:RequestSecurityTokenResponse>
  </S:Body>
</S:Envelope>
```

Note the following in the Response message:

- Since the SAML token doesn't support references using URI fragments (XML Id), an attached reference is returned that can be used *verbatim* to reference the token when it is placed inside a message.
- Since no explicit key type was specified in the request for the returned token, the IP/STS generates a symmetric key as the proof key for the returned token.
- A display token containing textual representation of the actual token is returned.

5.3. Authenticating with KerberosV5 Service Ticket

The identity provider requires that the service requester submit a *Kerberosv5 service ticket* for the IP/STS as the credential to authenticate to the IP/STS when requesting security tokens.

5.3.1. Credential format

To enable the requester InfoCard system to obtain a Kerberosv5 service ticket for the IP/STS, the endpoint reference of the IP/STS specified in the InfoCard must include a "service principal name" claim as identity information under the `Identity` tag as defined in [Addressing-Ext]. The KDC in the appropriate domain/realm can identify the service account based on this information and issue the required Kerberosv5 service ticket. This would typically be used in enterprise intranet scenarios.

No explicit user credential needs to be specified in this case as it is implied by the Kerberos realm that the user logs into. The credential selector format for KerberosV5 defined in [InfoCard-Ref] has the following form:

```
<ic:KerberosV5Authenticate />
```

5.3.2. Security binding

The "symmetric" security binding as described by [WS-SecurityPolicy] should be used for this authentication method for the request and response messages.

To enable the requester InfoCard system to obtain a Kerberosv5 service ticket for the IP/STS, the endpoint reference of the IP/STS specified in the InfoCard must include a "service principal name" claim as identity information under the `Identity` tag as defined in [Addressing-Ext].

An example of the sugested policy assertions for this security binding is as follows (Note that the requester InfoCard system will retrieve the actual security policy from the IP/STS using WS-MetadataExchange to determine the security binding to use).

Endpoint policy:

```
<wsp:Policy>
  <sp:SymmetricBinding>
    <wsp:Policy>
      <sp:ProtectionToken>
        <wsp:Policy>
          <sp:KerberosToken sp:IncludeToken="…/IncludeToken/Once" />
```

```
        </wsp:Policy>
      </sp:ProtectionToken>
      <sp:AlgorithmSuite>
        <wsp:Policy>
          [Algorithm Suite assertion]
        </wsp:Policy>
      </sp:AlgorithmSuite>
      <sp:Layout>
        <wsp:Policy>
          <sp:Strict />
        </wsp:Policy>
      </sp:Layout>
      <sp:IncludeTimestamp />
      <!-- Policy assertions for other optional binding properties can
        appear here that affect the content and format of the messages
        exchanged with the IP/STS -->
    </wsp:Policy>
  </sp:SymmetricBinding>
</wsp:Policy>
```

5.3.3. Message exchange

For this security binding, message protection and security correlation for the request and response legs of the message exchange is provided by using the symmetric session key in the attached KerberosV5 service ticket. Message integrity and confidentiality is provided as per the message policy described earlier in Section 5.1. A request and response message based on the security policy specified in the previous section, and using an algorithm suite policy assertion `sp:Basic128` as defined by [WS-SecurityPolicy], is illustrated below.

Requester to IP/STS request message:

```
<S:Envelope ...>
   <S:Header>
    <wsa:Action wsu:Id="_1">
      http://schemas.xmlsoap.org/ws/2004/04/security/trust/RST/Issue
    </wsa:Action>
    <wsa:MessageID wsu:Id="_2">
      uuid:eb9e1c77-0cea-4f2f-a586-78c15536137c
    </wsa:MessageID>
    <wsa:To wsu:Id="_3">
      http://contoso.com/IP/STS
    </wsa:To>
    <wsa:ReplyTo wsu:Id="_4">
      <wsa:Address>
        http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
      </wsa:Address>
    </wsa:ReplyTo>
    <wsse:Security S:mustUnderstand="1">
      <wsu:Timestamp wsu:Id="_6">
        <wsu:Created>2004-10-18T09:02:00Z</wsu:Created>
        <wsu:Expires>2004-10-18T09:12:00Z</wsu:Expires>
      </wsu:Timestamp>
      <!-- Kerberosv5 service ticket as authentication token -->
      <wsse:BinarySecurityToken wsu:Id="_30"
        ValueType="http://www.docs.oasis-open.org/wss/2004/07/oasis-000000-
wss-kerberos-token-profile-1.0#Kerberosv5_AP_REQ"
        EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">
```

```
          MIIEZzCCA9CgAwIBAgIQEmtJZc0==
      </wsse:BinarySecrityToken>
      <!-- List of encrypted elements in the message per
           message confidentiality policy -->
      <xenc:ReferenceList>
         <xenc:DataReference URI="#_20" />
      </xenc:ReferenceList>
      <!—Signature using the Kerberosv5 service ticket  -->
      <ds:Signature wsu:Id=”_33”>
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
              Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
          <ds:SignatureMethod
              Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
          <ds:Reference URI="#_6">...</ds:Reference>
          <ds:Reference URI="#_1">...</ds:Reference>
          <ds:Reference URI="#_2">...</ds:Reference>
          <ds:Reference URI="#_3">...</ds:Reference>
          <ds:Reference URI="#_4">...</ds:Reference>
          <ds:Reference URI="#_10">...</ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>1XTzjSMVXx/r0e+6v4o0Gwf60WY=</ds:SignatureValue>
        <ds:KeyInfo>
          <wss:SecurityTokenReference>
            <wss:Reference URI="#_30"
              ValueType=" http://www.docs.oasis-open.org/wss/2004/07/oasis-
000000-wss-kerberos-token-profile-1.0#Kerberosv5_AP_REQ"/>
          </wss:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </S:Header>
  <S:Body wsu:Id=”_10”>
    <!-- Start encrypted Content
    <wst:RequestSecurityToken>
      <wst:TokenType>
          urn:oasis:names:tc:SAML:1.0:assertion
      </wst:TokenType>
      <wst:RequestType>
          http://schemas.xmlsoap.org/ws/2004/04/security/trust/Issue
      </wst:RequestType>
      <wsp:AppliesTo>
          ... endpoint reference of relying party ...
      </wsp:AppliesTo>
      <ic:InfoCardReference>
        <ic:CardId> http://xyz.com/CardId/d795621fa01d454285f9</ic:CardId>
      </ic:InfoCardReference>
      <wst:Claims
          wst:Dialect=”http://schemas.microsoft.com/ws/2005/05/identity”>
        <ic:Claim
          URI=”http://.../ws/2005/05/identity/claims/givenname”/>
        <ic:Claim
          URI=”http://.../ws/2005/05/identity/claims/surname”/>
      </wst:Claims>
    </wst:RequestSecurityToken>
    End encrypted content -->
    <xenc:EncryptedData Id="_20">
```

```
      <xenc:EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc" />
      <ds:KeyInfo>
        <wsse:SecurityTokenReference>
          <wsse:Reference URI="#_30"
            ValueType="http://www.docs.oasis-open.org/wss/2004/07/oasis-
000000-wss-kerberos-token-profile-1.0#Kerberosv5_AP_REQ"/>
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>Thp8EqU0S+A4Qu+==</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </S:Body>
</S:Envelope>
```

Note the following in the request message:

- The ordering of items in the security header follows the most optimal layout for a receiver to process its contents (as described by [WS-SecurityPolicy]).
- The relying party identity in the form of its endpoint reference is communicated to the IP/STS via the `wsp:AppliesTo` element (the example shown assumes that the IP/STS has specified the policy assertion `ic:RequireAppliesTo` in its issuance policy in the InfoCard).
- The InfoCard reference (CardId) is included in the RST message element.
- The message is signed with the key in the KerberosV5 authentication token; but the token itself is NOT included in the message signature scope.
- Encrypted message elements are encrypted with the key in the Kerberosv5 token.
- References to the Kerberosv5 token, which is included in the message, is made using the `wsse:Reference` based direct reference to the token.

IP/STS to Requester response message:

```
<S:Envelope ...>
   <S:Header>
    <wsa:Action wsu:Id="_1">
      http://schemas.xmlsoap.org/ws/2004/04/security/trust/RSTR/Issue
    </wsa:Action>
    <wsa:RelatesTo wsu:Id="_2">
      uuid:eb9e1c77-0cea-4f2f-a586-78c15536137c
    </wsa:RelatesTo>
    <wsa:To wsu:Id="_3">
      http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
    </wsa:To>
    <wsse:Security S:mustUnderstand="1">
      <wsu:Timestamp wsu:Id="_6">
        <wsu:Created>2004-10-18T09:02:00Z</wsu:Created>
        <wsu:Expires>2004-10-18T09:12:00Z</wsu:Expires>
      </wsu:Timestamp>
      <!-- List of encrypted elements in the message per
           message confidentiality policy -->
      <xenc:ReferenceList>
         <xenc:DataReference URI="#_20" />
      </xenc:ReferenceList>
      <!—Signature using the Kerberosv5 service ticket  -->
```

```
      <ds:Signature wsu:Id=”_33”>
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
              Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
          <ds:SignatureMethod
              Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
          <ds:Reference URI="#_6">...</ds:Reference>
          <ds:Reference URI="#_1">...</ds:Reference>
          <ds:Reference URI="#_2">...</ds:Reference>
          <ds:Reference URI="#_3">...</ds:Reference>
          <ds:Reference URI="#_10">...</ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>1XTzjSMVXx/r0e+6v4o0Gwf60WY=</ds:SignatureValue>
        <ds:KeyInfo>
          <wss:SecurityTokenReference>
            <wsse:KeyIdentfier
              ValueType=" http://www.docs.oasis-open.org/wss/2004/07/oasis-
000000-wss-kerberos-token-profile-1.0#Kerberosv5_AP_REQ">
                1XTzjSMVXx/r0e+6v4o0Gwf60WY=
            </wsse:KeyIdentfier>
          </wss:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </S:Header>
  <S:Body wsu:Id=”_10”>
    <!-- Start encrypted Content
    <wst:RequestSecurityTokenResponse>
      <wst:TokenType>
          urn:oasis:names:tc:SAML:1.0:assertion
      </wst:TokenType>
      <wst:Lifetime>
        <wsu:Created>2004-10-18T09:02:00Z</wsu:Created>
        <wsu:Expires>2004-10-18T09:12:00Z</wsu:Expires>
      </wst:Lifetime>
      <wst:RequestedSecurityToken>
        <saml:Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion" ...>
          ...
        </saml:Assertion>
      </wst:RequestedSecurityToken>
      <wst:RequestedAttachedReference>
        <wsse:SecurityTokenReference>
          <wsse:KeyIdentifier ValueType="...#SAMLAssertionID">
             _SAMLAssertionID
          </wsse:KeyIdentifier>
        </wsse:SecurityTokenReference>
      </wst:RequestedAttachedReference>
      <wst:RequestedProofToken>
        <wst:BinarySecret>
          +amQ87tQFs67gkiet==
        </wst:BinarySecret>
      </wst:RequestedProofToken>
    </wst:RequestSecurityTokenResponse>
    End encrypted content -->
    <xenc:EncryptedData Id="_20">
      <xenc:EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc" />
```

```
      <ds:KeyInfo>
        <wsse:SecurityTokenReference>
          <wsse:KeyIdentfier
            ValueType=" http://www.docs.oasis-open.org/wss/2004/07/oasis-
000000-wss-kerberos-token-profile-1.0#Kerberosv5_AP_REQ">
              1XTzjSMVXx/r0e+6v4o0Gwf60WY=
          </wsse:KeyIdentfier>
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>Thp8EqU0S+A4Qu+==</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </S:Body>
</S:Envelope>
```

Note the following in the Response message:

- The ordering of items in the security header follows the most optimal layout for a receiver to process its contents (as described by [WS-SecurityPolicy]).
- Since the SAML token doesn't support references using URI fragments (XML Id), an “attached reference element” is returned that can be used *verbatim* to reference the token when it is placed inside a message.
- Since no explicit key type was specified in the request for the returned token, the IP/STS generates a symmetric key as the proof key for the returned token.
- Encrypted message elements are encrypted with the key in the Kerberosv5 token.
- References to the Kerberosv5 token, which is NOT included in the message, is made using the `wsse:KeyIdentifier` based indirect reference to the token.

5.4. Authenticating with Software Based X509 Certificate

The identity provider requires that the service requester submit an *X509v3 certificate* for the user, where the certificate and keys are in a software-based store (e.g. Microsoft base cryptographic storage provider), as the credential to authenticate to the IP/STS when requesting security tokens.

5.4.1. Credential format

To enable the requester InfoCard system to locate the right X509 certificate for use, the SHA1 hash of the entire certificate content should be included as a “thumbprint” in the credential description present in the InfoCard. The thumbprint search value can be used with the appropriate platform-specific APIs (e.g. CAPI2 on Windows) to locate the right certificate.

The credential selector format for software-based X509V3 certificate defined in [InfoCard-Ref] has the following form:

```
<!-- Select certificate based on certificate thumbprint -->
<ic:X509V3Authenticate>
  <ds:X509Data>
    <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/2004/xx/
oasis-2004xx-wss-soap-message-security-1.1#ThumbprintSHA1">
        ... Thp8EqU0S+A4Qu+==
    </wsse:KeyIdentifier>
  </ds:X509Data>
</ic:X509V3Authenticate>
```

5.4.2. Security binding

The “asymmetric” security binding as described by [WS-SecurityPolicy] should be used for this authentication method for the request and response messages.

To enable the requester InfoCard system to obtain the security token for the IP/STS to be used in securing the message exchanges, the endpoint reference of the IP/STS specified in the InfoCard must include a X509 certificate as identity information for the endpoint under the `Identity` tag as defined in [Addressing-Ext].

An example of the sugested policy assertions for this security binding is as follows (Note that the requester InfoCard system will retrieve the actual security policy from the IP/STS using WS-MetadataExchange to determine the security binding to use).

Endpoint policy (Asymmetric binding):

```
<wsp:Policy>
  <sp:AsymmetricBinding>
    <wsp:Policy>
      <sp:RecipientToken>
        <wsp:Policy>
          <!-- The IP/STS certificate is obtained from the
            identity extension of the endpoint reference included
            in the InfoCard -->
          <sp:X509V3Token sp:IncludeToken=”.../IncludeToken/Always” />
        </wsp:Policy>
      </sp:RecipientToken>
      <sp:InitiatorToken>
        <wsp:Policy>
          <!-- The user’s credential is a X509 certificate obtained
            by using the credential selector criteria specified in
            the InfoCard -->
          <sp:X509V3Token sp:IncludeToken=”.../IncludeToken/Always” />
        </wsp:Policy>
      </sp:InitiatorToken>
      <sp:AlgorithmSuite>
        <wsp:Policy>
          [Algorithm Suite assertion]
        </wsp:Policy>
      </sp:AlgorithmSuite>
      <sp:Layout>
        <wsp:Policy>
          <sp:Strict />
        </wsp:Policy>
      </sp:Layout>
      <sp:IncludeTimestamp />
      <!-- Policy assertions for other optional binding properties can
        appear here that affect the content and format of the messages
        exchanged with the IP/STS -->
    </wsp:Policy>
  </sp:AsymmetricBinding>
</wsp:Policy>
```

As an alternative, the IP/STS could use the “transport” security binding as described earlier in Section 5.2.2 requiring the user’s authentication token (*i.e.* the X509v3 certificate) to be submitted as an endorsing supporting token in the RST request [WS-SecurityPolicy]. An example of the suggested policy assertions for this security binding is as follows.

Endpoint policy (Transport binding):

```
<wsp:Policy>
  <sp:TransportBinding>
    <wsp:Policy>
      <sp:TransportToken>
        <wsp:Policy>
          <sp:HttpsToken />
        </wsp:Policy>
      </sp:TransportToken>
      <sp:AlgorithmSuite>
        <wsp:Policy>
          [Algorithm Suite assertion]
        </wsp:Policy>
      </sp:AlgorithmSuite>
      <sp:Layout>
        <wsp:Policy>
          <sp:Strict />
        </wsp:Policy>
      </sp:Layout>
      <sp:IncludeTimestamp />
      <sp:EndorsingSupportingTokens>
        <wsp:Policy>
          <!-- The user's credential is a X509 certificate obtained
            by using the credential selector criteria specified in
            the InfoCard -->
          <sp:X509V3Token sp:IncludeToken="".../IncludeToken/Once" />
        </wsp:Policy>
      </sp:EndorsingSupportingTokens>
    </wsp:Policy>
  </sp:TransportBinding>
</wsp:Policy>
```

5.4.3. Message exchange

For this security binding, message protection and security correlation for the request and response legs of the message exchange is provided by using the symmetric session key in the attached KerberosV5 service ticket. Message integrity and confidentiality is provided as per the message policy described earlier in Section 5.1. A request and response message based on the security policy specified in the previous section, and using an algorithm suite policy assertion `sp:Basic128` as defined by [WS-SecurityPolicy], is illustrated below.

For the "asymmetric" security binding above, message protection and security correlation for the request and response legs of the message exchange is provided by two separate asymmetric keys – one from the user's X509 certificate used for authentication, and the other from the X509 certificate representing the identity information for the IP/STS endpoint. Message integrity and confidentiality is provided as per the message policy described earlier in Section 5.1. A request and response message based on the "asymmetric" security binding specified in the previous section, and using an algorithm suite policy assertion `sp:Basic128` as defined by [WS-SecurityPolicy], is illustrated below.

Requester to IP/STS request message:

```
<S:Envelope ...>
   <S:Header>
    <wsa:Action wsu:Id="_1">
      http://schemas.xmlsoap.org/ws/2004/04/security/trust/RST/Issue
    </wsa:Action>
    <wsa:MessageID wsu:Id="_2">
      uuid:eb9e1c77-0cea-4f2f-a586-78c15536137c
```

```
    </wsa:MessageID>
    <wsa:To wsu:Id="_3">
      http://contoso.com/IP/STS
    </wsa:To>
    <wsa:ReplyTo wsu:Id="_4">
      <wsa:Address>
        http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
      </wsa:Address>
    </wsa:ReplyTo>
    <wsse:Security S:mustUnderstand="1">
      <wsu:Timestamp wsu:Id="_6">
        <wsu:Created>2004-10-18T09:02:00Z</wsu:Created>
        <wsu:Expires>2004-10-18T09:12:00Z</wsu:Expires>
      </wsu:Timestamp>

      <!-- X509 certificate of PS/STS endpoint as recipient token -->
      <wsse:BinarySecurityToken wsu:Id="_40"
        ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0#X509v3"
        EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">
          MIIEZzCCA9CgAwIBAgIQEmtJZc0==
      </wsse:BinarySecrityToken>

      <!-- Symmetric key used for message confidentiality encrypted
           to the X509 certificate for the IP/STS endpoint  -->
      <!-- Start encrypted Content
      101110010011101001111010100011010101
      End encrypted content -->
      <xenc:EncryptedKey Id="_43">
        <xenc:EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p" />
        <ds:KeyInfo>
          <wss:SecurityTokenReference>
            <wss:Reference URI="#_40"
              ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0#X509v3" />
          </wss:SecurityTokenReference>
        </ds:KeyInfo>
        <xenc:CipherData>
          <xenc:CipherValue>LKalk8as7eE==</xenc:CipherValue>
        </xenc:CipherData>
        <!-- List of encrypted elements in the message per
             message confidentiality policy -->
        <xenc:ReferenceList>
         <xenc:DataReference URI="#_20" />
        </xenc:ReferenceList>
      </xenc:EncryptedKey>

      <!-- X509 certificate of the user as the initiator token
        (i.e. message signature token) -->
      <wsse:BinarySecurityToken wsu:Id="_30"
        ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0#X509v3"
        EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">
          N725hzCCA9CgAwIBAgIQEmtJZc0==
```

```
      </wsse:BinarySecrityToken>

      <!-- Message signature using the user's X509 certificate  -->
      <ds:Signature wsu:Id="_33">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
              Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
          <ds:SignatureMethod
              Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
          <ds:Reference URI="#_6">...</ds:Reference>
          <ds:Reference URI="#_1">...</ds:Reference>
          <ds:Reference URI="#_2">...</ds:Reference>
          <ds:Reference URI="#_3">...</ds:Reference>
          <ds:Reference URI="#_4">...</ds:Reference>
          <ds:Reference URI="#_10">...</ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>1XTzjSMVXx/r0e+6v4o0Gwf60WY=</ds:SignatureValue>
        <ds:KeyInfo>
          <wss:SecurityTokenReference>
            <wss:Reference URI="#_30"
              ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0#X509v3" />
          </wss:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </S:Header>
  <S:Body wsu:Id="_10">
    <!-- Start encrypted Content
    <wst:RequestSecurityToken>
      <wst:TokenType>
          urn:oasis:names:tc:SAML:1.0:assertion
      </wst:TokenType>
      <wst:RequestType>
          http://schemas.xmlsoap.org/ws/2004/04/security/trust/Issue
      </wst:RequestType>
      <wsp:AppliesTo>
          ... endpoint reference of relying party ...
      </wsp:AppliesTo>
      <ic:InfoCardReference>
        <ic:CardId>http://xyz.com/CardId/d795621fa01d454285f9</ic:CardId>
      </ic:InfoCardReference>
      <wst:Claims
          wst:Dialect="http://schemas.microsoft.com/ws/2005/05/identity">
        <ic:Claim
          URI="http://.../ws/2005/05/identity/claims/givenname"/>
        <ic:Claim
          URI="http://.../ws/2005/05/identity/claims/surname"/>
      </wst:Claims>
    </wst:RequestSecurityToken>
    End encrypted content -->
    <xenc:EncryptedData Id="_20">
      <xenc:EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc" />
      <xenc:CipherData>
        <xenc:CipherValue>Thp8EqU0S+A4Qu+==</xenc:CipherValue>
      </xenc:CipherData>
```

```
    </xenc:EncryptedData>
  </S:Body>
</S:Envelope>
```

Note the following in the request message:

- The ordering of items in the security header follows the most optimal layout for a receiver to process its contents (as described by [WS-SecurityPolicy]).
- The relying party identity in the form of its endpoint reference is communicated to the IP/STS via the `wsp:AppliesTo` element (the example shown assumes that the IP/STS has specified the policy assertion `ic:RequireAppliesTo` in its issuance policy in the InfoCard).
- The InfoCard reference (CardId) is included in the RST message element.
- The message is signed with the key in the X509 certificate authentication token of the user; but the token itself is NOT included in the message signature scope.
- A symmetric encryption key is generated to encrypt message elements, and the encryption key itself is encrypted with the X509 certificate of the IP/STS.
- References to the X509 certificates, which are included in the message as "binary security tokens," are made using the `wsse:Reference` based direct reference to the tokens.

IP/STS to Requester response message:

```
<S:Envelope ...>
   <S:Header>
    <wsa:Action wsu:Id="_1">
      http://schemas.xmlsoap.org/ws/2004/04/security/trust/RSTR/Issue
    </wsa:Action>
    <wsa:RelatesTo wsu:Id="_2">
      uuid:eb9e1c77-0cea-4f2f-a586-78c15536137c
    </wsa:RelatesTo>
    <wsa:To wsu:Id="_3">
      http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
    </wsa:To>
    <wsse:Security S:mustUnderstand="1">
      <wsu:Timestamp wsu:Id="_6">
        <wsu:Created>2004-10-18T09:02:00Z</wsu:Created>
        <wsu:Expires>2004-10-18T09:12:00Z</wsu:Expires>
      </wsu:Timestamp>

      <!-- X509 certificate of the user as recipient token -->
      <wsse:BinarySecurityToken wsu:Id="_40"
        ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0#X509v3"
        EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">
          MIIEZzCCA9CgAwIBAgIQEmtJZc0==
      </wsse:BinarySecrityToken>

      <!-- Symmetric key used for message confidentiality encrypted
           to the X509 certificate of the user  -->
      <!-- Start encrypted Content
      10111001001110100111101010001101010 1
      End encrypted content -->
      <xenc:EncryptedKey Id="_43">
```

```
        <xenc:EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p" />
        <ds:KeyInfo>
          <wss:SecurityTokenReference>
            <wss:Reference URI="#_40"
              ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0#X509v3" />
          </wss:SecurityTokenReference>
        </ds:KeyInfo>
        <xenc:CipherData>
          <xenc:CipherValue>LKalk8as7eE==</xenc:CipherValue>
        </xenc:CipherData>
        <!-- List of encrypted elements in the message per
             message confidentiality policy -->
        <xenc:ReferenceList>
         <xenc:DataReference URI="#_20" />
        </xenc:ReferenceList>
      </xenc:EncryptedKey>

      <!-- X509 certificate of IP/STS endpoint as initiator token
        (message signature token) -->
      <wsse:BinarySecurityToken wsu:Id="_30"
        ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0#X509v3"
        EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-soap-message-security-1.0#Base64Binary">
          N725hzCCA9CgAwIBAgIQEmtJZc0==
      </wsse:BinarySecrityToken>

      <!-- Message signature using the IP/STS endpoint X509 certificate  -->
      <ds:Signature wsu:Id=”_33”>
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
              Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
          <ds:SignatureMethod
              Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
          <ds:Reference URI="#_6">...</ds:Reference>
          <ds:Reference URI="#_1">...</ds:Reference>
          <ds:Reference URI="#_2">...</ds:Reference>
          <ds:Reference URI="#_3">...</ds:Reference>
          <ds:Reference URI="#_10">...</ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>1XTzjSMVXx/r0e+6v4o0Gwf60WY=</ds:SignatureValue>
        <ds:KeyInfo>
          <wss:SecurityTokenReference>
            <wss:Reference URI="#_30"
              ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0#X509v3" />
          </wss:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </S:Header>
  <S:Body wsu:Id=”_10”>
    <!-- Start encrypted Content
    <wst:RequestSecurityTokenResponse>
      <wst:TokenType>
```

```
          urn:oasis:names:tc:SAML:1.0:assertion
      </wst:TokenType>
      <wst:Lifetime>
        <wsu:Created>2004-10-18T09:02:00Z</wsu:Created>
        <wsu:Expires>2004-10-18T09:12:00Z</wsu:Expires>
      </wst:Lifetime>
      <wst:RequestedSecurityToken>
        <saml:Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion" ...>
          ...
        </saml:Assertion>
      </wst:RequestedSecurityToken>
      <wst:RequestedAttachedReference>
        <wsse:SecurityTokenReference>
          <wsse:KeyIdentifier ValueType="...#SAMLAssertionID">
             _SAMLAssertionID
          </wsse:KeyIdentifier>
        </wsse:SecurityTokenReference>
      </wst:RequestedAttachedReference>
      <wst:RequestedProofToken>
        <wst:BinarySecret>
          +amQ87tQFs67gkiet==
        </wst:BinarySecret>
      </wst:RequestedProofToken>
    </wst:RequestSecurityTokenResponse>
    End encrypted content -->
    <xenc:EncryptedData Id="_20">
      <xenc:EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc" />
      <xenc:CipherData>
        <xenc:CipherValue>Thp8EqU0S+A4Qu+==</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </S:Body>
</S:Envelope>
```

Note the following in the Response message:

- The ordering of items in the security header follows the most optimal layout for a receiver to process its contents (as described by [WS-SecurityPolicy]).
- Since the SAML token doesn't support references using URI fragments (XML Id), an “attached reference element” is returned that can be used *verbatim* to reference the token when it is placed inside a message.
- Since no explicit key type was specified in the request for the returned token, the IP/STS generates a symmetric key as the proof key for the returned token.
- The message is signed with the key in the X509 certificate of the IP/STS endpoint; but the token itself is NOT included in the message signature scope.
- A symmetric encryption key is generated to encrypt message elements, and the encryption key itself is encrypted with the X509 certificate authentication token of the user.
- References to the X509 certificates, which are included in the message as “binary security tokens,” are made using the `wsse:Reference` based direct reference to the tokens.

5.5. Authenticating with Smartcard Based X509 Certificate

The identity provider requires that the service requester submit an *X509v3 certificate* for the user, where the certificate and keys are in a hardware-based smart card, as the credential to authenticate to the IP/STS when requesting security tokens.

5.5.1. Credential format

To enable the requester InfoCard system to locate the right X509 certificate for use, the SHA1 hash of the entire certificate content should be included as a "thumbprint" in the credential description present in the InfoCard. The thumbprint search value can be used with the appropriate platform-specific APIs (e.g. CAPI2 on Windows) to locate the right certificate. Additionally, a user hint is included in the `ic:CredentialHint` element as part of the credential type specification which can be used to prompt the user for inserting the appropriate smart card into the reader.

The credential selector format for hardware-based X509V3 certificate defined in [InfoCard-Ref] has the following form:

```
<!-- Provide hint and select certificate based on certificate thumbprint -->
<ic:CredentialHint>
    Please insert your corporate smart card
</ic:CredentialHint>
<ic:X509V3Authenticate>
  <ds:X509Data>
    <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/2004/xx/
oasis-2004xx-wss-soap-message-security-1.1#ThumbprintSHA1">
        ... Thp8EqU0S+A4Qu+==
    </wsse:KeyIdentifier>
  </ds:X509Data>
</ic:X509V3Authenticate>
```

5.5.2. Security binding

The "asymmetric" security binding as described by [WS-SecurityPolicy] should be used for this authentication method for the request and response messages.

To enable the requester InfoCard system to obtain the security token for the IP/STS to be used in securing the message exchanges, the endpoint reference of the IP/STS specified in the InfoCard must include a X509 certificate as identity information for the endpoint under the `Identity` tag as defined in [Addressing-Ext].

An example of the suggested policy assertions for this security binding is identical to that given in the earlier Section 5.4.2. (Note that the requester InfoCard system will retrieve the actual security policy from the IP/STS using WS-MetadataExchange to determine the security binding to use).

As an alternative, the IP/STS could use the "transport" security binding as described earlier in Section 5.2.2 requiring the user's authentication token (*i.e.* the X509v3 certificate) to be submitted as an endorsing supporting token in the RST request [WS-SecurityPolicy]. An example of the suggested policy assertions for this security binding is given earlier in Section 5.4.2.

5.5.3. Message exchange

The message exchanges for the "asymmetric" binding are identical to those in Section 5.4.3.

5.6. Authenticating with Self-issued Token

The identity provider requires that the service requester submit a *self-issued SAML token* carrying the asymmetric public key of the user as the credential to authenticate to the IP/STS when requesting security tokens. The self-issued token and key are provided by the self-issued identity provider of the InfoCard system.

5.6.1. Credential format

To enable the requester InfoCard system to locate the right self-issued InfoCard for use as the authentication credential, the `ic:PrivatePersonalIdentifier` claim used to identify the user at the IP/STS must be specified as the credential descriptor in the InfoCard. The requester InfoCard system can locate the appropriate self-issued InfoCard which is associated with the specified private personal identifier.

The credential selector format for self-issued token defined in [InfoCard-Ref] has the following form:

```
<ic:SelfIssuedAuthenticate>
  <ic:PrivatePersonalIdentifier>
      xqh78FgyuThp8EqU0S+A4Qu+=
  </ic:PrivatePersonalIdentifier>
</ic:SelfIssuedAuthenticate>
```

5.6.2. Security binding

The “symmetric” or “asymmetric” security binding as described by [WS-SecurityPolicy] should be used for this authentication method depending on the *type of key* requested in the SAML token produced by the self-issued identity provider.

To enable the requester InfoCard system to obtain the security token for the IP/STS to be used in securing the message exchanges, the endpoint reference of the IP/STS specified in the InfoCard must include a X509 certificate as identity information for the endpoint under the `Identity` tag as defined in [Addressing-Ext].

An example of the suggested policy assertions for the “symmetric” security binding is as follows (Note that the requester InfoCard system will retrieve the actual security policy from the IP/STS using WS-MetadataExchange to determine the security binding to use).

Endpoint policy (Symmetric binding):

```
<wsp:Policy>
  <sp:SymmetricBinding>
    <wsp:Policy>
      <sp:ProtectionToken>
        <wsp:Policy>
          <sp:IssuedToken sp:IncludeToken=”.../IncludeToken/Once”>
            <sp:Issuer>
              <wsa:EndpointReference>
               <wsa:Address>
                http://schemas.microsoft.com/ws/2005/05/identity/issuer/self
               </wsa:Address>
              </wsa:EndpointReference>
            </sp:Issuer>
            <sp:RequestSecurityTokenTemplate>
              <wst:TokenType>urn:oasis:names:tc:SAML:1.1</wst:TokenType>
              <wst:KeyType>
                http://schemas.xmlsoap.org/ws/2004/04/trust/SharedKey
              </wst:KeyType>
```

```
            </sp:RequestSecurityTokenTemplate>
          </sp:IssuedToken>
        </wsp:Policy>
      </sp:ProtectionToken>
      <sp:AlgorithmSuite>
        <wsp:Policy>
          [Algorithm Suite assertion]
        </wsp:Policy>
      </sp:AlgorithmSuite>
      <sp:Layout>
        <wsp:Policy>
          <sp:Strict />
        </wsp:Policy>
      </sp:Layout>
      <sp:IncludeTimestamp />
      <!-- Policy assertions for other optional binding properties can
        appear here that affect the content and format of the messages
        exchanged with the IP/STS -->
    </wsp:Policy>
  </sp:SymmetricBinding>
</wsp:Policy>
```

An example of the suggested policy assertions for an “asymmetric” security binding is as follows (Note that the requester InfoCard system will retrieve the actual security policy from the IP/STS using WS-MetadataExchange to determine the security binding to use).

Endpoint policy (Asymmetric binding):

```
<wsp:Policy>
  <sp:AsymmetricBinding>
    <wsp:Policy>
      <sp:RecipientToken>
        <wsp:Policy>
          <!-- The IP/STS certificate is obtained from the
            identity extension of the endpoint reference included
            in the InfoCard -->
          <sp:X509V3Token sp:IncludeToken=”.../IncludeToken/Always” />
        </wsp:Policy>
      </sp:RecipientToken>
      <sp:InitiatorToken>
        <wsp:Policy>
          <!-- The user’s credential is a self-issued SAML token -->
          <sp:IssuedToken sp:IncludeToken=”.../IncludeToken/Once”>
            <sp:Issuer>
              <wsa:EndpointReference>
               <wsa:Address>
                http://schemas.microsoft.com/ws/2005/05/identity/issuer/self
               </wsa:Address>
              </wsa:EndpointReference>
            </sp:Issuer>
            <sp:RequestSecurityTokenTemplate>
              <wst:TokenType>urn:oasis:names:tc:SAML:1.1</wst:TokenType>
              <wst:KeyType>
                http://schemas.xmlsoap.org/ws/2004/04/trust/PublicKey
              </wst:KeyType>
            </sp:RequestSecurityTokenTemplate>
          </sp:IssuedToken>
        </wsp:Policy>
```

```
      </sp:InitiatorToken>
      <sp:AlgorithmSuite>
        <wsp:Policy>
          [Algorithm Suite assertion]
        </wsp:Policy>
      </sp:AlgorithmSuite>
      <sp:Layout>
        <wsp:Policy>
          <sp:Strict />
        </wsp:Policy>
      </sp:Layout>
      <sp:IncludeTimestamp />
      <!-- Policy assertions for other optional binding properties can
        appear here that affect the content and format of the messages
        exchanged with the IP/STS -->
    </wsp:Policy>
  </sp:AsymmetricBinding>
</wsp:Policy>
```

As an alternative, the IP/STS could use the “transport” security binding as described earlier in Section 5.2.2 requiring the user’s authentication token (*i.e.* the SAML token) to be submitted as an endorsing supporting token in the RST request [WS-SecurityPolicy]. An example of the suggested policy assertions for this security binding is as follows.

Endpoint policy (Transport binding):

```
<wsp:Policy>
  <sp:TransportBinding>
    <wsp:Policy>
      <sp:TransportToken>
        <wsp:Policy>
          <sp:HttpsToken />
        </wsp:Policy>
      </sp:TransportToken>
      <sp:AlgorithmSuite>
        <wsp:Policy>
          [Algorithm Suite assertion]
        </wsp:Policy>
      </sp:AlgorithmSuite>
      <sp:Layout>
        <wsp:Policy>
          <sp:Strict />
        </wsp:Policy>
      </sp:Layout>
      <sp:IncludeTimestamp />
      <sp:EndorsingSupportingTokens>
        <wsp:Policy>
          <!-- The user’s credential is a self-issued SAML token -->
          <sp:IssuedToken sp:IncludeToken=”.../IncludeToken/Once”>
            <sp:Issuer>
              <wsa:EndpointReference>
               <wsa:Address>
                http://schemas.microsoft.com/ws/2005/05/identity/issuer/self
               </wsa:Address>
              </wsa:EndpointReference>
            </sp:Issuer>
            <sp:RequestSecurityTokenTemplate>
              <wst:TokenType>urn:oasis:names:tc:SAML:1.1</wst:TokenType>
```

```
              <wst:KeyType>
                http://schemas.xmlsoap.org/ws/2004/04/trust/SharedKey
              </wst:KeyType>
            </sp:RequestSecurityTokenTemplate>
          </sp:IssuedToken>
        </wsp:Policy>
      </sp:EndorsingSupportingTokens>
    </wsp:Policy>
  </sp:TransportBinding>
</wsp:Policy>
```

5.6.3. Message Exchange

For the “symmetric” security binding, message protection and security correlation for the request and response legs of the message exchange is provided by using the symmetric proof key in the attached self-issued SAML token. Message integrity and confidentiality is provided as per the message policy described earlier in Section 5.1. A request and response message based on the security policy specified in the previous section, and using an algorithm suite policy assertion `sp:Basic128` as defined by [WS-SecurityPolicy], is illustrated below. This example also illustrates the use of the `wst:UseKey` element and the corresponding security header elements when requesting an asymmetric key token.

Requester to IP/STS request message:

```
<S:Envelope ...>
   <S:Header>
    <wsa:Action wsu:Id="_1">
      http://schemas.xmlsoap.org/ws/2004/04/security/trust/RST/Issue
    </wsa:Action>
    <wsa:MessageID wsu:Id="_2">
      uuid:eb9e1c77-0cea-4f2f-a586-78c15536137c
    </wsa:MessageID>
    <wsa:To wsu:Id="_3">
      http://contoso.com/IP/STS
    </wsa:To>
    <wsa:ReplyTo wsu:Id="_4">
      <wsa:Address>
        http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
      </wsa:Address>
    </wsa:ReplyTo>
    <wsse:Security S:mustUnderstand="1">
      <wsu:Timestamp wsu:Id="_6">
        <wsu:Created>2004-10-18T09:02:00Z</wsu:Created>
        <wsu:Expires>2004-10-18T09:12:00Z</wsu:Expires>
      </wsu:Timestamp>

      <!-- Self-issued SAML token as authentication token encrypted
           to symmetric encryption key K which in turn is encrypted
           to the X509 certificate of the IP/STS endpoint  -->
      <!-- Start encrypted Content
      01001010101000101010100101010010101010010
      End encrypted content -->
      <xenc:EncryptedKey>
        <xenc:EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p" />
        <ds:KeyInfo>
          <ds:X509Data>
            <wsse:KeyIdentifier
```

```
              ValueType="http://docs.oasis-open.org/wss/2004/xx/oasis-2004xx-
wss-soap-message-security-1.1#ThumbprintSHA1">
                EdFoIaAeja85201XTzjNMVWy7532jUYtrx=
            </wsse:KeyIdentifier>
          </ds:X509Data>
        </ds:KeyInfo>
        <xenc:CipherData>
          <xenc:CipherValue>Ukasdj8257Fjwf=</xenc:CipherValue>
        </xenc:CipherData>
      </xenc:EncryptedKey>
      <!-- Start encrypted Content
      <saml:Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion" ...
            AssertionID="_SelfIssuedAssertionID" ...>
        ...
      </saml:Assertion>
      End encrypted content -->
      <xenc:EncryptedData>
        <xenc:EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc" />
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
          <wsse:KeyIdentfier
            ValueType="http://docs.oasis-open.org/wss/2004/xx/oasis-2004xx-
wss-soap-message-security-1.1#EncryptedKey">
              SdFoIaAeja85201XTzjSMVXx=
          </wsse:KeyIdentfier>
            </wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
        <xenc:CipherData>
          <xenc:CipherValue>aKlh4817JerpZoDofy90=</xenc:CipherValue>
        </xenc:CipherData>
      </xenc:EncryptedData>

      <!-- List of encrypted elements in the message per
           message confidentiality policy -->
      <xenc:ReferenceList>
         <xenc:DataReference URI="#_20" />
      </xenc:ReferenceList>

      <!—Message signature using the self-issued token  -->
      <ds:Signature wsu:Id=”_33”>
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
              Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
          <ds:SignatureMethod
              Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
          <ds:Reference URI="#_6">...</ds:Reference>
          <ds:Reference URI="#_1">...</ds:Reference>
          <ds:Reference URI="#_2">...</ds:Reference>
          <ds:Reference URI="#_3">...</ds:Reference>
          <ds:Reference URI="#_4">...</ds:Reference>
          <ds:Reference URI="#_10">...</ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>1XTzjSMVXx/r0e+6v4o0Gwf60WY=</ds:SignatureValue>
        <ds:KeyInfo>
          <wss:SecurityTokenReference>
```

```
            <wsse:KeyIdentfier
              ValueType="...#SAMLAssertionID">
                _SelfIssuedAssertionID
            </wsse:KeyIdentfier>
          </wss:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>

      <ds:Signature Id="_proofKeySignature">
        <!-- signature proving possession of the requested public key -->
        <!-- as proof key; KeyInfo in signature contains the public key -->
        <KeyInfo>
          <KeyValue>
            <RSAKeyValue>
              <Modulus>...</Modulus>
              <Exponent>...</Exponent>
            </RSAKeyValue>
          </KeyValue>
        </KeyInfo>
      </ds:Signature>

    </wsse:Security>
  </S:Header>
  <S:Body wsu:Id=”_10”>
    <!-- Start encrypted Content
    <wst:RequestSecurityToken>
      <wst:TokenType>
          urn:oasis:names:tc:SAML:1.0:assertion
      </wst:TokenType>
      <wst:RequestType>
          http://schemas.xmlsoap.org/ws/2004/04/security/trust/Issue
      </wst:RequestType>
      <wst:KeyType>
        http://schemas.xmlsoap.org/ws/2004/04/trust/PublicKey
      </wst:KeyType>
      <ic:InfoCardReference>
        <ic:CardId> http://xyz.com/CardId/d795621fa01d454285f9</ic:CardId>
      </ic:InfoCardReference>
      <wst:Claims
          wst:Dialect=”http://schemas.microsoft.com/ws/2005/05/identity”>
        <ic:Claim
          URI=”http://.../ws/2005/05/identity/claims/givenname”/>
        <ic:Claim
          URI=”http://.../ws/2005/05/identity/claims/surname”/>
      </wst:Claims>
      <wst:UseKey Sig="#_proofKeySignature" />
    </wst:RequestSecurityToken>
    End encrypted content -->
    <xenc:EncryptedData Id="_20">
      <xenc:EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc" />
      <ds:KeyInfo>
        <wsse:SecurityTokenReference>
          <wsse:KeyIdentfier
            ValueType="...#SAMLAssertionID">
              _SelfIssuedAssertionID
          </wsse:KeyIdentfier>
```

```
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>Thp8EqU0S+A4Qu+==</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </S:Body>
</S:Envelope>
```

Note the following in the request message:

- The ordering of items in the security header follows the most optimal layout for a receiver to process its contents (as described by [WS-SecurityPolicy]).
- Since an asymmetric key token is being requested, the proof key along with the corresponding proof of possession for the private key is included via the `wst:UseKey` element.
- The InfoCard reference (`CardId`) is included in the RST message element.
- The message is signed with the proof key in the self-issued SAML authentication token; but the token itself is NOT included in the message signature scope.
- Encrypted message elements are encrypted with the proof key in the self-issued SAML token.
- References to the self-issued SAML token, which is included in the message, is made using the using the `wsse:KeyIdentifier` based indirect reference to the token.

IP/STS to Requester response message:

```
<S:Envelope ...>
   <S:Header>
    <wsa:Action wsu:Id="_1">
      http://schemas.xmlsoap.org/ws/2004/04/security/trust/RSTR/Issue
    </wsa:Action>
    <wsa:RelatesTo wsu:Id="_2">
      uuid:eb9e1c77-0cea-4f2f-a586-78c15536137c
    </wsa:RelatesTo>
    <wsa:To wsu:Id="_3">
      http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
    </wsa:To>
    <wsse:Security S:mustUnderstand="1">
      <wsu:Timestamp wsu:Id="_6">
        <wsu:Created>2004-10-18T09:02:00Z</wsu:Created>
        <wsu:Expires>2004-10-18T09:12:00Z</wsu:Expires>
      </wsu:Timestamp>
      <!-- List of encrypted elements in the message per
           message confidentiality policy -->
      <xenc:ReferenceList>
         <xenc:DataReference URI="#_20" />
      </xenc:ReferenceList>
      <!-- Signature using the self-issued SAML token  -->
      <ds:Signature wsu:Id=”_33”>
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
              Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
          <ds:SignatureMethod
              Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
          <ds:Reference URI="#_6">...</ds:Reference>
```

```
        <ds:Reference URI="#_1">...</ds:Reference>
        <ds:Reference URI="#_2">...</ds:Reference>
        <ds:Reference URI="#_3">...</ds:Reference>
        <ds:Reference URI="#_10">...</ds:Reference>
      </ds:SignedInfo>
      <ds:SignatureValue>1XTzjSMVXx/r0e+6v4o0Gwf60WY=</ds:SignatureValue>
      <ds:KeyInfo>
        <wsse:SecurityTokenReference>
          <wsse:KeyIdentfier
            ValueType="...#SAMLAssertionID">
              _SelfIssuedAssertionID
          </wsse:KeyIdentfier>
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
    </ds:Signature>
  </wsse:Security>
</S:Header>
<S:Body wsu:Id=”_10”>
  <!-- Start encrypted Content
  <wst:RequestSecurityTokenResponse>
    <wst:TokenType>
        urn:oasis:names:tc:SAML:1.0:assertion
    </wst:TokenType>
    <wst:Lifetime>
      <wsu:Created>2004-10-18T09:02:00Z</wsu:Created>
      <wsu:Expires>2004-10-18T09:12:00Z</wsu:Expires>
    </wst:Lifetime>
    <wst:RequestedSecurityToken>
      <saml:Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion" ...>
        ...
      </saml:Assertion>
    </wst:RequestedSecurityToken>
    <wst:RequestedAttachedReference>
      <wsse:SecurityTokenReference>
        <wsse:KeyIdentifier ValueType="...#SAMLAssertionID">
           _SAMLAssertionID
        </wsse:KeyIdentifier>
      </wsse:SecurityTokenReference>
    </wst:RequestedAttachedReference>
  </wst:RequestSecurityTokenResponse>
  End encrypted content -->
  <xenc:EncryptedData Id="_20">
    <xenc:EncryptionMethod
        Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc" />
    <ds:KeyInfo>
      <wsse:SecurityTokenReference>
        <wsse:KeyIdentfier
          ValueType="...#SAMLAssertionID">
            _SelfIssuedAssertionID
        </wsse:KeyIdentfier>
      </wsse:SecurityTokenReference>
    </ds:KeyInfo>
    <xenc:CipherData>
      <xenc:CipherValue>Thp8EqU0S+A4Qu+==</xenc:CipherValue>
    </xenc:CipherData>
  </xenc:EncryptedData>
</S:Body>
```

```
</S:Envelope>
```

Note the following in the Response message:

- The ordering of items in the security header follows the most optimal layout for a receiver to process its contents (as described by [WS-SecurityPolicy]).
- Since the SAML token doesn't support references using URI fragments (XML Id), an “attached reference element” is returned that can be used *verbatim* to reference the token when it is placed inside a message.
- Since an asymmetric key token was requested and an ephemeral public was supplied as the proof key in the request, the response message does not include an explicit proof token.
- Encrypted message elements are encrypted to the proof key in the self-issued token.
- References to the self-issued SAML token, which is NOT included in the response message, are made using the `wsse:KeyIdentifier` based indirect reference.

6. Faults

In addition to the standard faults described in WS-Addressing, WS-Security and WS-Trust, the InfoCard Technical Reference [InfoCard-Ref] defines additional faults that may be generated by the relying party or the identity provider.

7. References

[InfoCard-Ref]
"A Technical Reference for InfoCard v1.0 in Windows," August 2005.

[SOAP 1.2]
M. Gudgin, et al, "SOAP Version 1.2 Part 1: Messaging Framework," June 2003.

[WS-Addressing]
D. Box et al, "Web Services Addressing (WS-Addressing)," August 2004.

[Addressing-Ext]
Document to be published in the near future.

[WS-MetadataExchange]
"Web Services Metadata Exchange (WS-MetadataExchange)," August 2004.

[WS-Security]
A. Natalin et al, “Web Services Security: SOAP Message Security 1.0”, May 2004.

[WS-Policy]
"Web Services Policy Framework (WS-Policy)," September 2004.

[WS-SecurityPolicy]
"Web Services Security Policy Language (WS-SecurityPolicy)," July 2005.

[WS-Trust]
"Web Services Trust Language (WS-Trust)," May 2004.

[XMLDSIG]
Eastlake III, D., Reagle, J., and Solo, D., “XML-Signature Syntax and Processing”, http://www.ietf.org/rfc/rfc3275.txt, March 2002.

[XMLENC]
Imamura, T., Dillaway, B., and Simon, E., “XML Encryption Syntax and Processing”, http://www.w3.org/TR/xmlenc-core/, August 2002.

[XML Schema, Part 1]
H. Thompson et al, "XML Schema Part 1: Structures," May 2001.

[XML Schema, Part 2]
P. Biron et al, "XML Schema Part 2: Datatypes," May 2001.

Appendix A – Self-Issued Identity Provider

The InfoCard system in the Windows client platform includes a simple identity provider called the “Self-issued Identity Provider” (see Figure 1) to allow users to self-assert identity in the form of self-issued security tokens. This may be acceptable, for example, when accessing a retail bookseller Web service to set up an account. The retail service may allow the user to self-assert her own name and address information as claims in a security token that is self-signed.

This section describes how a relying party that consumes self-issued tokens can evaluate and use them. Note that an identity provider can also be the relying party for self-issued tokens if it accepts a self-issued token as the authentication credential for a user during security token requests. This is described in more detail in Section 5.6.

Self-issued Token Characteristics

The characteristics of a self-issued token, including its format and the supported claim types, are described in [InfoCard-Ref].

Following is an example of a self-issued security token containing three attributes (or claims) and using an asymmetric proof key.

Example:

```
<Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
    AssertionID="uuid-08301dba-d8d5-462f-85db-dec08c5e4e17"
    Issuer="http://schemas.microsoft.com/ws/2005/05/identity/issuer/self"
    IssueInstant="2004-10-06T16:44:20.00Z"
    MajorVersion="1" MinorVersion="1">
  <Conditions NotBefore="2004-10-06T16:44:20.00Z"
    NotOnOrAfter="2004-10-06T16:49:20.00Z"/>
  <AttributeStatement>
    <Subject>
      <SubjectConfirmation>
        <ConfirmationMethod>
          urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
        </ConfirmationMethod>
        <ds:KeyInfo>
          <!-- The proof key goes here. The content of this element
            is either a symmetric key or a RSA public key depending
            on what is specified by the relying party -->
          <!-- asymmetric RSA public key as proof key -->
          <KeyValue>
           <RSAKeyValue>
             <Modulus>...</Modulus>
             <Exponent>AQAB</Exponent>
           </RSAKeyValue>
          </KeyValue>
        </ds:KeyInfo>
      </SubjectConfirmation>
    </Subject>
    <Attribute AttributeName=”privatpersonalidentifier”
      AttributeNamespace=”http://schemas.microsoft.com/ws/2005/05/identity”>
      <AttributeValue>
        f8301dba-d8d5a904-462f0027-85dbdec0
      </AttributeValue>
    </Attribute>
```

```
    <Attribute AttributeName=”givenname”
      AttributeNamespace=”http://schemas.microsoft.com/ws/2005/05/identity”>
      <AttributeValue>dasf</AttributeValue>
    </Attribute>
    <Attribute AttributeName=”emailaddress”
      AttributeNamespace=”http://schemas.microsoft.com/ws/2005/05/identity”>
      <AttributeValue>dasf@mail.com</AttributeValue>
    </Attribute>
  </AttributeStatement>
  <Signature xmlns=”http://www.w3.org/2000/09/xmldsig#”>
    <SignedInfo>
      <CanonicalizationMethod
        Algorithm=”http://www.w3.org/2001/10/xml-exc-c14n#”/>
      <SignatureMethod
        Algorithm=”http://www.w3.org/2000/09/xmldsig#rsa-sha1”/>
      <Reference URI=”uuid-08301dba-d8d5-462f-85db-dec08c5e4e17”>
        <Transforms>
          <Transform
            Algorithm=” http://.../2000/09/xmldsig#enveloped-signature”/>
          <Transform
            Algorithm=”http://www.w3.org/2001/10/xml-exc-c14n#”/>
        </Transforms>
        <DigestMethod
          Algorithm=”http://www.w3.org/2000/09/xmldsig#sha1”/>
        <DigestValue>vpnIyEi4R/S4b+1vEH4gwQ9iHsY=</DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue>...</SignatureValue>
    <KeyInfo>
      <KeyValue>
        <RSAKeyValue>
          <Modulus>...</Modulus>
          <Exponent>AQAB</Exponent>
        </RSAKeyValue>
      </KeyValue>
    </KeyInfo>
  </Signature>
</Assertion>
```

Note the following in the self-issued token shown above:

- The issuer of the token, indicated by the value of the `saml:Issuer` attribute, is specified as the URI *http://schemas.microsoft.com/ws/2005/05/identity/issuer/self*.
- The subject confirmation key (or proof key) included in the issued token is a RSA public key within the `saml:SubjectConfirmation` element (see the first block of highlighted text in the token above).
- The signature key included in the issued token is specified as a RSA public key within the token signature (see the second block of highlighted text in the token above).

Consuming Self-issued Tokens

A relying party can accept and consume self-issued tokens from the user where that is convenient and appropriate. As mentioned earlier, an identity provider can also be the relying party for self-issued tokens. It can accept a self-issued token, which has been registered with it by the user earlier, as the authentication credential for a user during security token requests. In addition to the characteristics of self-issued tokens described in

[InfoCard-Ref], a consumer of self-issued tokens should also adhere to the following guidelines.

- The RSA public key specified in the signature of the self-issued token should be used as the long-term key associated with the user issuing the self-issued token.
- When accepting and verifying a self-issued token, ensure that the current time falls within the token's validity interval.
- The subject confirmation key (proof key) specified in the self-issued token should be treated as a short-term key that can be exercised by the service requester during the token validity interval to demonstrate that it is the subject of the token.
- If a long-term unique identifier for the user is needed (*e.g.* the relying party needs an identifier for its local representation of the user where profile information about the user is kept), then ask for the "Private Personal Identifier" claim in the self-issued token in policy (see description of that claim in [InfoCard-Ref]). This claim provides a privacy friendly identifier for the principal that is the subject of the security token.

Appendix B – Glossary

Authentication

Authentication is the process of validating security credentials.

Claim

A claim is a statement made about an entity such as a sender, a service, or other resource (e.g., name, identifier, key, group, privilege, capability, etc.). It is sometimes referred to as an *assertion* in the security literature.

Claims Authority

A claims authority is an entity that can authenticate principals and make specific claims about them which other services may trust. For example, an authority may assert a user's name, address and social security number as claims which another service may trust and accept. A claims authority is typically a security token service.

Principal

A principal is any entity about which claims can be made by an identity provider. These entities include users, services, computers and devices.

Security token

A security token represents a collection of one or more claims.

Security Token Service

A Security Token Service (STS) is a Web service that issues security tokens – that is, the service makes assertions – based on evidence that it trusts, to those services that trust the service or to specific recipients.

Signed security token

A signed security token is a security token that is cryptographically endorsed by a specific authority (e.g., an X509 certificate, a Kerberos ticket or a SAML assertion).